Răzvan Rughiniș

Computer Science

Proiectarea rețelelor

Editura PRINTECH

Material cu drept de autor

1 Arhitectura unui ruter

1.1 Componentele unui ruter

Dacă ne reprezentăm un ruter ca un dispozitiv similar unui calculator, nu vom fi deloc departe de realitate. Simplificând, un ruter este un dispozitiv ce realizează interconectarea între două sau mai multe rețele. Prin intermediul ruterelor, pachetele de date pot traversa granițele rețelelor, parcurgând oricâte rețele este necesar pentru a ajunge la destinație. De fapt, necesitatea de rutare a pachetelor a existat dinaintea construirii primului ruter ca dispozitiv dedicat acestei sarcini. Funcția principală a unui ruter este aceea de a ruta pachete sau, altfel spus, de a executa un algoritm ce are drept scop determinarea căii pe care un pachet de rețea trebuie să o urmeze până la destinație.

Adevărata utilitate a unui dispozitiv de rutare este demonstrată în momentul în care acesta interconectează cel puțin două rețele. O modalitate ușor accesibilă de a realiza o funcție de rutare între două rețele este aceea ce a folosi un calculator cu două plăci de rețea, conectate la rețele diferite. Toate sistemele majore de operare pentru sisteme desktop (Windows, Linux, Mac OS) suportă, deseori nativ, funcții de rutare. Spre exemplu, ICS (*Internet Connection Sharing*) din sistemele Windows reprezintă funcția de rutare. Pe sistemele Linux, rutarea se activează prin setarea variabilei `ip_forward` din fișierul `/proc/sys/net/ipv4`.

Folosirea unui calculator drept înlocuitor pentru un ruter este posibilă, ușor de implementat, dar deseori ineficientă. Performanța traficului între rețele (și în Internet, deopotrivă) depinde într-o mare măsură de performanța ruterelor pe care pachetele le traversează. Timpul pe care un ruter îl folosește pentru a lua deciziile de rutare și pentru a executa funcțiile diverselor servicii pe care le rulează reprezintă cea mai semnificativă întârziere în traficul de rețea.

Ruterele, pe de altă parte, sunt dispozitive concepute special pentru a realiza cât mai eficient funcțiile de rutare precum și alte servicii necesare în rețea. Din acest motiv, arhitectura hardware a ruterelor este considerabil diferită de cea a unui calculator desktop, spre exemplu. Cu toate acestea, un ruter deține o serie de componente ce se regăsesc în interiorul oricărui calculator: un procesor (CPU) aritmetico-logic, o memorie volatilă (RAM), precum și zone de stocare permanentă a datelor (ROM, NVRAM, *Flash*). De asemenea, ruterul comunică în exterior prin interfețe iar întreaga sa funcționalitate este controlată de un sistem de operare ce rulează în permanență.

1.1.1 Memoriile unui ruter

1.1.1.1 Memoria RAM

Ca şi în cazul calculatoarelor uzuale, memoria RAM este folosită pentru a stoca instrucțiunile şi datele ce participă în operațiile executate de procesor la un moment dat. În cazul ruterelor Cisco, în timpul secvenței de inițializare întregul conținut al sistemului de operare este încărcat în memoria RAM. Dacă sistemul de operare prezintă capabilități extinse (servicii sau protocoale suplimentare), în mod implicit consumul său de memorie va fi mai ridicat.

Modificarea parametrilor unui ruter, deci configurarea sa prin intermediul sistemului de operare, are ca efect păstrarea acestor modificări într-un fişier special localizat în RAM, ce stochează configurația curentă, denumită şi *running configuration* sau *running-config*. În afara cazului în care configurația unui ruter este salvată într-o memorie permanentă, aceasta se pierde împreună cu întregul conținut al memoriei RAM la repornirea ruterului.

Pe lângă sistemul de operare şi configurația curentă, memoria RAM stochează o multitudine de variabile, structuri de date, statistici şi alte date temporare utilizate de serviciile ce rulează pe ruterul respectiv. Memoria RAM stochează tabela de rutare a unui ruter, aceasta fiind o bază de date cu toate destinațiile cunoscute de ruter, folosită pentru rutarea pachetelor. Printre alte informații stocate în memoria RAM se numără cache-ul ARP ce păstrează asocierile adresă MAC – adresă IP pentru toate rețelele Ethernet la care este conectat ruterul, statistici ale pachetelor şi stărilor conexiunilor, precum şi mesaje de log.

1.1.1.2 Memoria ROM

Memoria ROM este o formă de memorie non-volatilă, de dimensiuni foarte mici. Ea stochează în permanență instrucțiunile necesare inițializării ruterului: activarea diferitelor module, efectuarea de teste hardware, etc. Instrucțiunile (sau programul) de inițializare mai poartă denumirea de *bootstrap*.

De asemenea, memoria ROM stochează instrucțiuni primare pentru diagnostice, utile mai ales în cazul funcționării anormale a ruterului, precum şi o versiune minimală a sistemului de operare (numit *ROMMonitor* sau *rommon*) ce permite, printre altele, înlocuirea sistemului de operare instalat pe un ruter.

Software-ul din memoria ROM poartă denumirea de *firmware* şi, de regulă, nu necesită rescrieri sau actualizări.

1.1.1.3 Memoria Flash

Din moment ce memoria *Flash* este memoria non-volatilă cu cea mai mare dimensiune dintr-un ruter, principalul ei scop este acela de a păstra în permanență o copie nealterată a sistemului de operare. La inițializarea unui ruter, sistemul de operare este citit din memoria *Flash*, eventual decomprimat şi încărcat în memoria RAM.

Conceptual, memoria *Flash* reprezintă echivalentul unui hard disk, cu un sistem de fişiere propriu ce permite stocarea şi altor fişiere pe lângă sistemul de operare propriu-zis. Mai mult chiar, o memorie *Flash* suficient de mare poate stoca mai multe sisteme de operare.

Memoria *Flash* este deseori implementată sub forma unor module SIMM sau plăci PCMCIA ce pot fi uşor înlocuite fără a deschide fizic carcasa ruterului.

1.1.1.4 Memoria NVRAM

Memoria NVRAM este o memorie permanentă de mici dimensiuni folosită de ruter pentru a stoca fişierul de configurare ce va fi încărcat în memoria RAM după iniţializarea sistemului de operare. Configuraţia unui ruter poate fi salvată în NVRAM pentru a fi păstrată şi încărcată la fiecare pornire a ruterului. Fişierul comenzilor de configurare din NVRAM poartă numele de *startup configuration* sau *startup-config*.

Memoria NVRAM poate fi rescrisă de oricâte ori este necesară o salvare a configuraţiei curente.

1.1.2 Interfeţele unui ruter

1.1.2.1 Porturi de management

Spre deosebire de celelalte interfeţe, porturile de management nu sunt folosite pentru transportarea traficului de reţea ci doar pentru administrare. Ele nu necesită o configuraţie IP şi pot fi accesate printr-o conexiune directă de la portul serial al unui calculator, folosind un cablu de tip *crossover*.

Deşi accesarea sistemului de operare al unui ruter poate fi făcută prin orice interfaţă a sa, atât de management cât şi de reţea, configurarea iniţială a unui ruter trebuie făcută printr-un port de management.

Portul de consolă oferă administratorului o interfaţă în linie de comandă (CLI – *Command Line Interface*) pentru accesarea sistemului de operare şi introducerea comenzilor de configurare. Aceeaşi interfaţă poate fi accesată şi prin Telnet sau SSH pe un ruter configurat corespunzător, dar astfel de conexiuni, ce implică trafic de pachete, trebuie făcute prin interfeţele de reţea[1].

1.1.2.2 Interfeţele de reţea

În general, o interfaţă de reţea a unui ruter reprezintă un conector fizic ce suportă trafic de pachete. Interfeţele pot fi conectate la o multitudine de medii de transmisie (cupru, fibră optică, wireless), fiecare având diferite standarde, ceea ce creează o varietate considerabilă de tipuri de interfeţe.

Interfeţele sunt considerate conexiuni spre reţele diferite şi pot fi configurate cu adrese IP şi măşti de reţea corespunzătoare. Sistemul de operare al ruterului nu va permite configurarea a două interfeţe diferite cu aceeaşi pereche de parametri de adresă IP şi mască de reţea, ceea ce ar cauza apartenenţa a două interfeţe la aceeaşi reţea. Interfeţele de reţea pot accepta atât pachete IP destinate către ruterul local, cât şi pachete cu alte destinaţii, ce vor fi rutate în funcţie de informaţiile din tabela de rutare.

Interfeţele unui ruter sunt identificate în mod unic printr-un nume, ce desemnează tipul lor, urmat de unul sau mai multe numere ce indică numărul lor şi, respectiv, modulul instalat pe ruter din care aceasta face parte. Spre exemplu, Ethernet0 desemnează prima (0) interfaţă Ethernet a unui ruter, în timp ce Serial 1/2 indică a treia interfaţă serială (2) a celui de-al doilea modul (1). Identificatorii numerici sunt consecutivi la nivel de modul şi interfaţă. Denumirile de interfeţe în aceste forme sunt utilizate atât în rezultatele comenzilor de interogare cât şi în construcţia comenzilor, ca parametri.

Conform denumirii lor, **interfeţele LAN** sunt folosite pentru a conecta ruterul la reţele locale Ethernet, similar cu modul în care un calculator se conectează la o reţea locală prin intermediul

---

[1] Mai mult chiar, unele rutere şi switchuri suportă şi configurarea prin intermediul unei interfeţe web sau a unor aplicaţii specializate, precum *Cisco Security Device Manager* (SDM).

plăcii sale de rețea. Din punctul de vedere al rețelei, interfața LAN se comportă ca orice altă stație direct atașată segmentului de rețea respectiv: interfața deține o adresă unică MAC de nivel 2, şi trimite cereri ARP precum şi răspunsuri la acestea cu adresa sa configurată de nivel 3.

O interfață Ethernet foloseşte un conector de tip RJ-45 ce suportă cablare UTP (*Unshielded Twisted Pair*). Din punctul de vedere al tipului de cablu folosit, interfața se comportă asemenea unei plăci de rețea, deci va necesita un cablu *crossover* la conectarea cu un PC sau un alt ruter sau un cablu *straight-through* la conectarea cu un port al unui switch.

Interfețele WAN sunt folosite pentru conexiuni cu rețele întinse pe arii geografice mai mari. Diferitele tipuri de interfețe WAN se datorează în principal diferitelor tipuri de încapsulări de nivel 2 suportate, cum sunt HDLC (*High-Level Data Link Control*), PPP, sau *Frame Relay*. Pentru rețele WAN, adresarea la nivelul legătură de date se face prin metode proprii fiecărui protocol, adresarea MAC fiind folosită doar în rețele Ethernet.

1.1.3 Procesul de inițializare al unui ruter

Stările prin care un ruter trece în timpul inițializării sale, imediat după pornire, pot fi rezumate în următoarele etape:

Efectuarea POST (*Power-On Self Test*) reprezintă un proces de execuție a unor instrucțiuni de test cu scopul de a determina funcționarea corespunzătoare a tuturor componentelor hardware. POST este rulați cu scopul de a detecta hardware -ul instalat în ruter: capacitatea memoriilor, prezența interfețelor și tipul lor, etc. O mare parte a echipamentelor (nu doar ruterele) folosesc acest procedeu la inițializare iar instrucțiunile sunt stocate de regulă în memoria ROM.

Execuția programului Bootstrap are drept scop localizarea imaginii sistemului de operare ce va fi încărcat în RAM. Programul de *Bootstrap* este, de asemenea, localizat în memoria ROM.

Încărcarea sistemului de operare se realizează, de cele mai multe ori, prin copierea imaginii sistemului de operare din *Flash* în memoria RAM. Imaginea mai poate fi încărcată și prin rețea, de pe un server TFTP (*Trivial File Transfer Protocol*), în cazul în care ruterului i se specifică explicit această metodă sau în situația în care nu este găsită o imagine validă în *Flash*. Încărcarea imaginii poate fi făcută în paralel cu decompresia ei, pentru imagini stocate în formă arhivată. Unele rutere pot rula imaginea sistemului de operare direct din memoria *Flash*.

Lipsa unei imagini a sistemului de operare va avea ca rezultat încărcarea unui sistem de operare minimal, direct din ROM, oferind posibilitatea administratorului de a instala o imagine corespunzătoare.

Încărcarea fișierului de configurare se realizează în vederea configurării sistemului de operare încărcat. Acesta nu conține decât o configurație de bază, implicită, ce nu permite niciun fel de trafic prin interfețele sale. Fișierul de configurare (*startup-config*) poate conține comenzile necesare pentru configurarea interfețelor ruterului, a rutelor, parolelor, serviciilor suplimentare și a oricăror alți parametri suportați de ruter. Fișierul de configurare este căutat în memoria NVRAM și încărcat în RAM, unde va fi referit ca *running-config* (configurația curentă). Comenzile încărcate din fișierul de configurare au efect imediat.

Dacă memoria NVRAM este vidă, ruterul va căuta să încarce o configurație de pe un server TFTP. Dacă nu va fi găsit niciun server TFTP, ruterul va oferi utilizatorului, prin interfața de administrare, accesul la *setup mode*, adică posibilitatea să răspundă la o serie de întrebări de configurare, introducând astfel o configurație minimală în *running-config*.

Dacă administratorul este conectat la consola ruterului, procesul prezentat mai sus poate fi observat începând cu finalizarea cu succes a POST-ului.

1.2 Sistemul de operare

Sistemul de operare produs de către Cisco pentru echipamentele sale (routere și switchuri deopotrivă) este numit **Cisco IOS** (*Internetwork Operating System*) și cuprinde toate funcțiile de rutare, switching, securitate și monitorizare, precum și o varietate de servicii ce determină capabilitățile echipamentelor ce îl rulează.

Cisco IOS este disponibil într-o multitudine de versiuni, fiecare beneficiind de capabilități orientate spre anumite platforme hardware. Facilitățile pot include: protocoale, algoritmi de criptare, suport pentru anumite tipuri de interfețe etc. Sistemele de operare sunt mapate pe anumite serii de modele de routere și switchuri atât din punctul de vedere al capabilităților fiecărei platforme cât și datorită limitărilor hardware specifice seriilor de modele (încărcarea memoriei RAM, gradul de utilizare a procesorului).

Interfața caracteristică pentru Cisco IOS este linia de comandă, în mod text, accesibilă prin conexiuni directe la porturile de management precum și prin protocoale ca Telnet sau SSH. Comenzile de configurare și monitorizare sunt introduse în mod text și sunt grupate în moduri ce stabilesc contextul valid pentru fiecare comandă. Spre exemplu, un anumit set de comenzi va fi disponibil în modul de configurare a unei interfețe Ethernet, iar alt set va putea fi accesat în modul de configurare al serviciului DHCP de pe un ruter.

Similar sistemelor Linux, comenzile pot fi executate în moduri privilegiate corespunzătoare. Cisco IOS permite configurarea de până la 15 nivele de privilegii, precum și implementarea de conturi de utilizator asociate acestor nivele. Nivelul inferior al acestor privilegii permite doar executarea unui set limitat de comenzi, din gama comenzilor de monitorizare, în timp ce nivelul superior permite executarea oricărei comenzi de configurare și monitorizare, indiferent de efectul potențial fatal al acestora.

La momentul scrierii acestei cărți cea mai recentă versiune a Cisco IOS este denumită 12.4(24)T. Notația are următoarea semnificație:

- 12 reprezintă versiunea majoră;
- 4 reprezintă versiunea minoră, iar împreună cu versiunea majoră formează partea de *maintenance release* a versiunii;
- 24 reprezintă numărul *release*-ului, urmat uneori de un număr de *release* interimar.
- T reprezintă un „*train*", adică un set de capabilități orientate spre un anumit segment de piață. Spre exemplu, „T" reprezintă „*Technology*", „S" reprezintă „*Service provider*", șamd. „*Train*"-urile sunt excepții de la varianta „*mainline*", cea mai stabilă dar cu setul cel mai restrâns de capabilități.

1.2.1 Moduri de configurare

Modurile de configurare din Cisco IOS sunt similare unor contexte și determină ce comenzi sunt valide și pot fi introduse la un moment dat.

Prima clasificare a modurilor de configurare ale unui ruter se face din prisma nivelurilor de securitate:

- Modul **User EXEC** permite doar execuția comenzilor de interogare a stării ruterului;
- Modul **Privileged EXEC** permite și execuția comenzilor de modificare a configurației unui ruter.

Modul accesat în mod implicit în urma conectării la un ruter este modul *User EXEC*, indicat în linia de comandă printr-un prompt format din numele ruterului (**Router**, în mod implicit) urmat de caracterul „>". Pentru trecerea în modul privilegiat se folosește comanda **enable** iar pentru revenirea în modul *User EXEC*, comanda **disable**. Trecerea în modul privilegiat poate fi protejată printr-o parolă și este indicată de simbolul „#" atașat numelui ruterului:

```
Router con0 is now available.

Press RETURN to get started.

Router>enable
Router#disable
Router>
```

În modurile *User EXEC* și *Privileged EXEC* pot fi executate comenzi de interogare a ruterului, dar nu pot fi executate comenzi ce modifică configurația curentă. Pentru modificarea fișierului de configurare din memorie este necesară intrarea într-un mod special de configurare, accesibil doar din modul privilegiat, prin comanda **configure terminal**. Modul se mai numește și **modul global de configurare** deoarece din el derivă toate celelalte moduri de configurare.

```
Router#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#enable password cisco
Router(config)#exit
Router#disable
Router>enable
Password: [administratorul introduce parola corectă: "cisco"]
Router#
```

În exemplul de mai sus se observă că intrarea în modul de configurare este reflectată în modificarea prompt-ului, în **Router(config)#**. Comanda **enable password** configurează ruterul să interogheze utilizatorul pentru o parolă validă (**cisco**, în exemplul de față) înainte de a permite accesul la modul privilegiat (denumit uneori și modul *„enable"* datorită comenzii).

Modurile de configurare în Cisco IOS sunt indicate prin modificarea prompt-ului și formează o ierarhie având modul **(config)** la bază. Pentru revenirea la modul imediat superior se poate folosi comanda **exit**. De asemenea, comanda **end** va întoarce întotdeauna prompt-ul până la nivelul *Privileged EXEC*, indiferent de adâncimea nivelului de configurare curent[1].

Următorul exemplu demonstrează navigarea prin modurile de configurare al unui protocol de rutare (**config-router**), respectiv al unei interfețe (**config-if**):

```
Router#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#router rip
Router(config-router)#exit
Router(config)interface Ethernet0
Router(config-if)end
Router#
```

1.2.2 Sistemul de fișiere

1.3 Configurații de bază

Numele oricărui ruter este, în mod implicit *„Router"* și este vizibil în orice prompt al liniei de comandă. Deși numele are semnificație exclusiv locală, păstrarea numelor implicite pe toate

[1] Pentru rapiditate se poate folosi și combinația de taste CTRL-Z, echivalentă comenzii `end`.

echipamentele poate crea ambiguități chiar și într -o rețea formată din câteva rutere. Schimbarea numelui ruterului se face din modul global de configurare, prin comanda **hostname**:

```
Router#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#hostname Marketing
Marketing(config)#
```

Un alt exemplu îl reprezintă setarea datei și orei ceasului intern al ruterului, utile în principal pentru identificarea momentelor de timp ale diverselor evenimente, atât în timp real cât și în jurnalele (*log*-urile) de sistem. Un exemplu de setare a datei este următorul[1]:

```
Marketing#clock set 01:44:00 20 September 2009
Marketing#
Marketing#show clock
01:44:07.495 UTC Sun Sep 20 2009
```

După cum se observă, comanda **clock set** nu a fost introdusă în modul global de configurare, deși comanda **show clock** arată că aceasta a fost executată corect. Acest lucru se datorează faptului că toate comenzile introduse în modul global de configurare au drept efect modificarea configurației curente din memoria RAM a ruterului (*running-config*). Această configurație poate fi salvată în orice moment în memoria NVRAM pentru a fi încărcată la fiecare repornire a ruterului. În consecință, setarea datei la valoarea introdusă ca argument al comenzii clock set la fiecare încărcare a configurației nu ar avea sens. Comanda acționează doar asupra procesului ce gestionează ceasul ruterului și doar o singură dată, la momentul execuției.

Comanda **show clock** este una dintre numeroasele comenzi de interogare disponibile în linia de comandă a Cisco IOS. Comenzile de acest tip sunt rezervate pentru modurile *User EXEC* și *Privileged EXEC*, nefiind permise în niciunul dintre modurile de configurare.

Configurația curentă poate fi afișată prin comanda **show running-config**. Atât *running-config* cât și *startup-config* au o structură care păstrează comenzile într-o formă asemănătoare celei în care au fost introduse, grupate după modurile corespunzătoare. Această structură facilitează copierea secvențelor de comenzi dintr -un fișier de configurare și introducerea lor direct la terminalul utilizat.

```
Marketing#show running-config
Building configuration...

Current configuration : 1466 bytes
!
version 12.3
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname Marketing
!
boot-start-marker
boot-end-marker
!
!
no aaa new-model
ip subnet-zero
!
!
no ip domain lookup
[...]
```

[1] În practică se folosește NTP (*Network Time Protocol*) pentru setarea ceasului.

În mod similar se poate lista configurația salvată în NVRAM, folosind comanda **show startup-config**.

Salvarea configurației din RAM în NVRAM se realizează folsind comanda **copy**:

```
Marketing#copy running-config startup-config
Destination filename [startup-config]?
Building configuration...
[OK]
Marketing#
```

Aceeași comandă realizează și copierea comenzilor din *startup-config* în *running-config*, cu diferența că, în acest caz, comenzile din *startup-config* vor fi introduse peste cele din *running-config*, combinând efectiv cele două configurații. Comenzile multiple ce nu pot coexista (spre exemplu comanda **hostname**) se suprascriu, păstrând efectul celei mai recente comenzi, după cum e de așteptat, dar alte comenzi vor fi introduse peste cele deja prezente, putând să creeze efecte nedorite. Practic, comanda **copy startup-config running-config** este echivalentă copierii linie cu linie a fișierului din NVRAM în linia de comandă.

Pentru a evita situația de mai sus, se poate folosi comanda **configure replace nvram:**, care va înlocui complet configurația curentă cu cea din NVRAM. Doar versiunile mai recente de IOS suportă această comandă.

1.3.1 Protejarea accesului la ruter

În afara configurațiilor de rețea ce asigură funcționarea unui ruter, restricționarea accesului la un ruter reprezintă unul dintre primele obiective ce trebuie avute în vedere, pentru a reduce riscurile de securitate.

1.3.1.1 Protejarea modului privilegiat

După cum s-a arătat anterior, accesul la modul privilegiat de configurare poate (și ar trebui să fie) protejat printr-o parolă. Comanda **enable password** permite administratorului să specifice o parolă pentru acest scop, dar dezavantajul ei constă în faptul că această comandă, împreună cu parola dată drept argument, sunt stocate necriptat în fișierul de configurație, fiind vizibile la o simplă afișare a acestuia:

```
Marketing(config)#enable password cisco
Marketing(config)#exit
Marketing#show running-config
Building configuration...

Current configuration : 1570 bytes
!
! Last configuration change at 01:57:34 UTC Sun Sep 20 2009
!
version 12.3
hostname Marketing
!
enable password cisco
!
[...]
```

O primă variantă de securizare atât a parolei de *enable* cât și a altor parole stocate necriptat în fișierul de configurare, este activarea serviciului de criptare a parolelor prin comanda **service password encryption**. Algoritmul de criptare aplicat este unul bazat pe transpoziții și este extrem de ușor reversibil (există scripturi online care realizează decriptarea aproape instantaneu):

```
Marketing(config)#enable password cisco
```

```
Marketing(config)#service password-encryption
Marketing#show running-config
  [...]
!
hostname Marketing
!
 enable password 7 02050D480809
[...]
```

Algoritmul de criptare este denotat de numărul 7. Această criptare este uneori întâlnită în documentații sub numele de *„Cisco Type 7 Encryption"* și nu este considerată o măsură de securitate solidă.

Metoda cea mai sigură de stocare a parolei de *enable* este calcularea unui *hash* MD5 din textul parolei și stocarea acestuia în fișierul de configurare. La introducerea unei parole în prompt-ul de *login*, acesteia i se va aplica același algoritm și se vor compara *hash*-urile rezultate. În acest fel, parola nu este efectiv ținută, iar algoritmul MD5 asigură faptul că parola inițială nu poate fi dedusă din *hash*. Pentru a proteja parola în acest fel se folosește comanda **`enable secret`**, ca în exemplul de mai jos:

```
Marketing(config)#enable secret cisco
The enable secret you have chosen is the same as your enable password.
This is not recommended.  Re-enter the enable secret.

Marketing(config)#enable secret class
Marketing(config)#exit
Marketing#show running-config
Building configuration...
[...]
version 12.3
service password-encryption
!
hostname Marketing
!
enable secret 5 $1$pL39$AX6DY0RveVsZGtpUiOGfa1
enable password 7 02050D480809
```

Din comenzile anterioare se observă în primul rând faptul că parola configurată cu **`enable secret`** nu poate avea aceeași valoare cu cea configurată cu **`enable password`**, tocmai pentru că cea din urmă poate fi ușor dedusă. Deși cele două moduri de configurare a parolei nu sunt mutual exclusive în fișierul de configurare, în momentul în care comanda **`enable secret`** a fost introdusă, ruterul va ignora parola configurată în **`enable password`** și o va verifica doar pe cea din **`enable secret`**.

Fișierul de configurare reține doar *hash*-ul parolei și desemnează existența unui *hash* MD5 prin codul 5 introdus înaintea parolei. De asemenea, configurația curentă conține acum și comanda de activare a serviciului de criptare, **`service password-encryption`**.

1.3.1.2 Protejarea liniilor de acces la ruter

S-a menționat deja faptul că una dintre metodele de a accesa interfața sistemului de operare al unui ruter este conectarea direct la portul de consolă al acestuia. Consola nu este catalogată de către IOS ca o interfață deoarece nu participă în procesul de rutare a pachetelor, ci este considerată o „linie" de acces (*line*).

De asemenea, un ruter suportă conexiuni de la distanță, folosind protocoalele Telnet și SSH, care oferă un acces similar la linia de comandă a sistemului de operare. Aceste conexiuni sunt realizate, în ultimă instanță, prin una sau mai multe interfețe de rețea ale ruterului, configurate pentru traficul de pachete.

Ruterul gestionează o serie de terminale virtuale ce funcționează asemenea unor interfețe aflate la capetele conexiunilor Telnet sau SSH sosite prin oricare dintre interfețele fizice al ruterului. Terminalele virtuale poartă denumirea *vty* și sunt numerotate asemenea interfețelor. În mod implicit există 5 terminale, de la vty0 la vty4, dar pot fi configurate mai multe. În momentul în care un administrator se conectează prin Telnet sau SSH la un ruter, el va ocupa unul dintre aceste terminale virtuale atâta timp cât conexiunea sa TCP va fi activă.

Este de dorit ca, din moment ce ruterele sunt accesibile din (teoretic) orice rețea din topologia pe care o formează, sistemul lor de operare să nu poată fi accesat de către persoane neautorizate. Este recomandabil ca atât terminalele virtuale cât și consola să fie protejate prin parole. Aceste parole vor restricționa nu doar accesul la modul privilegiat, ci și la modul *User EXEC*, astfel că un utilizator care încearcă să acceseze linia de comandă fără o parolă valabilă nu va putea executa nicio comandă pe ruterul respectiv.

Configurarea parolelor pe toate liniile accesibile este exemplificată în cele ce urmează:

```
Marketing#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Marketing(config)#line vty 0 4
Marketing(config-line)#password cisco
Marketing(config-line)#login
Marketing(config-line)#exit
Marketing(config)#line console 0
Marketing(config-line)#password cisco
Marketing(config-line)#login
Marketing(config-line)#end
Marketing#
```

Se observă că liniile de acces au un mod de configurare separat, aflat imediat sub modul global de configurare.

Comanda **`line vty 0 4`** specifică faptul că noul mod de configurare va afecta toate cele 5 terminale virtuale (de la 0 la 4), iar comenzile **`password`** si **`login`** se vor aplica celor 5 linii simultan. Comanda **`login`** indică faptul că linia respectivă va cere o informație de autentificare înainte de a autoriza accesul. Configurarea comenzii **`password`** fără comanda **`login`** va avea ca efect omiterea autentificării iar configurarea comenzii **`login`** fără **`password`** va face autentificarea imposibilă (ruterul va aștepta la infinit o parolă inexistentă).

Mai jos este inclusă secțiunea corespunzătoare din fișierul de configurare. Parolele au fost criptate datorită comenzii **`service password-encryption`** introduse anterior:

```
Marketing#show running-config | begin line
line con 0
 exec-timeout 0 0
 password 7 01100F175804
 logging synchronous
 login
line aux 0
line vty 0 4
 password 7 104D000A0618
 login
!
```

1.3.1.3 Configurarea mesajelor de întâmpinare

Mesajele de întâmpinare (banner) reprezintă texte ce sunt afișate înainte sau după autentificarea prin o linie de acces. Scopurile lor sunt diverse și pot include notificări pentru alți administratori sau utilizatori, sau informații despre ruterul la care s-a încercat conectarea. În orice

caz, un banner minimal trebuie să specifice cel puțin faptul că sistemul respectiv nu este accesibil persoanelor neautorizate și astfel să nu „invite" potențialii atacatori.

MoTD, sau *Message of The Day* este un banner ce este afișat înaintea autentificării atât pe vty-uri cât și la consolă. Un astfel de banner se configurează în modul următor:

```
Marketing(config)#banner motd *
Enter TEXT message.  End with the character '*'.
Acces neautorizat!
Acest ruter este proprietatea companiei X.
*
Marketing(config)#end
Marketing#exit

Marketing con0 is now available

Press RETURN to get started.

Acces neautorizat!
Acest ruter este proprietatea companiei X.

User Access Verification

Password:
Marketing>
```

Banner-ele pot ocupa linii multiple, de aceea se folosește un caracter delimitator la începutul și sfârșitul mesajului, caracter ce nu trebuie să se regăsească în corpul său. Se observă momentul în care textul banner-ului este afișat, la conectarea prin consolă. Același banner este afișat și la conectarea la unul dintre terminalele virtuale.

1.3.2 Configurarea unei interfețe

Pentru configurarea interfețelor de rețea se folosește un mod specific de configurare, accesat prin comanda **`interface`** urmată de numele și numărul interfeței. Odată schimbat modul de configurare, toate comenzile ulterioare se vor aplica interfeței menționate ca argument, până la părăsirea modului.

Pentru ca o interfață să poată accepta pachete din rețeaua la care este conectată, configurația sa trebuie să conțină cel puțin o adresă IP și o mască de rețea. În plus, implicit, toate interfețele unui ruter sunt închise, deci o interfață trebuie activată înainte sau după ce i se atașează o adresă:

```
Marketing(config)#interface serial1/0
Marketing(config-if)#ip address 10.0.0.1 255.255.255.0
Marketing(config-if)#no shutdown
Sep 20 05:15:59.811: %LINK-3-UPDOWN: Interface Serial1/0, changed state to
up
Sep 20 05:16:00.815: %LINEPROTO-5-UPDOWN: Line protocol on Interface
Serial1/0, changed state to up
Marketing(config-if)#
```

Asignarea unei adrese IPv4 unei interfețe se face prin comanda **`ip address`**, urmată de adresa IP dorită și masca de rețea corespunzătoare. Ruterul nu va permite configurarea a două interfețe cu adrese ale căror subneturi se suprapun.

Interfețele neconfigurate au un set de comenzi implicite în configurația curentă, precum cel din exemplul următor:

```
Marketing#show running-config interface serial 1/2
Building configuration...

Current configuration : 76 bytes
!
```

```
interface Serial1/2
 no ip address
 shutdown
 serial restart-delay 0
end
```

Prefixul **no** înaintea comenzilor anulează efectul comenzii respective. Starea implicită a unei interfețe este **shutdown**, în care interfața nu participă sub nicio formă la traficul de rețea. Din această cauză, activarea unei interfețe presupune eliminarea acestei comenzi din configurație, prin comanda **no shutdown**. În mod similar, spre exemplu, se poate elimina o adresă configurată pe o anumită interfață, prin comanda **no ip address**, care este, de fapt, un parametru al stării implicite pentru interfețele neconfigurate.

```
Sep 20 05:15:59.811: %LINK-3-UPDOWN: Interface Serial1/0, changed state to
up
Sep 20 05:16:00.815: %LINEPROTO-5-UPDOWN: Line protocol on Interface
Serial1/0, changed state to up
```

Cele două mesaje returnate de ruter indică faptul că o conexiune punct la punct a fost stabilită prin interfața serială 1/0. Primul mesaj este emis ca rezultat al activării interfeței, indicând faptul că legătura la nivel fizic a fost stabilită, iar al doilea confirmă efectuarea legăturii la nivelul legătură de date *(data link)*. Cu alte cuvinte, protocolul (încapsularea) de la nivelul 2 a realizat adiacența cu interfața de la celălalt capăt.

O practică utilă în rețelistică, în special în rețelele cu mai mulți administratori, este menținerea unei evidențe a legăturilor între echipamente și a scopului pe care aceste legături le deservesc. Etichetarea cablurilor poate fi o primă etapă, dar Cisco IOS permite administratorilor introducerea de descrieri direct la nivelul interfețelor, folosind comanda **description**:

```
Marketing(config)#interface serial1/0
Marketing(config-if)#description Engineering Link
Marketing(config-if)#end
Marketing#show running interface serial 1/0
Building configuration...

Current configuration : 209 bytes
!
interface Serial1/0
 description Engineering Link
 ip address 10.0.0.1 255.255.255.0
end
```

1.3.2.1 Protocolul CDP

Un protocol extrem de util pentru verificarea legăturilor dintre echipamente Cisco este protocolul CDP (*Cisco Discovery Protocol*). CDP schimbă periodic pachete de informații cu vecinii săi, pe toate interfețele active ale ruterului. În acest fel, fiecare ruter poate identifica celelalte echipamente la care este conectat.

CDP este activ în mod implicit pe toate interfețele , dar nu poate schimba informații în ambele sensuri decât cu echipamente Cisco[1]. Prin faptul că CDP trimite implicit informații despre dispozitivul local pe toate interfețele, protocolul poate fi considerat o breșă de securitate deoarece oferă unui potențial atacator detalii importante despre echipamentele existente în rețea.

Spre exemplu, pentru legătura punct la punct configurată anterior, vecinul ruterului Marketing apare din perspectiva sa în felul următor, ca rezultat al comenzii **show cdp neibghors**:

[1] Protocoalele a căror funcționalitate depinde de echipamentele sau protocoalele unui anumit producător se numesc protocoale proprietare.

```
Marketing#show cdp neighbors
Capability Codes: R - Router, T - Trans Bridge, B - Source Route Bridge
                  S - Switch, H - Host, I - IGMP, r - Repeater

Device ID        Local Intrfce     Holdtme    Capability  Platform  Port ID
Engineering      Ser 1/0            179          R S       3640      Ser 1/0
```

Comanda afișează ca rezultat un tabel în care listează toți vecinii detectați care rulează, de asemenea, CDP. În cazul de față, ruterul local, Marketing a detectat prin interfața Serial1/0 (câmpul **Local Intrfce**) pe ruterul Engineering[1], a cărui interfață corespondentă este tot Serial 1/0 (**Port ID**). Ruterul Engineering este un model 3640 și are capabilități de ruter și switch.

Comanda poate fi completată cu parametrul **detail**, care va afișa toate informațiile cunoscute despre vecini:

```
Marketing#show cdp neighbors detail
-------------------------
Device ID: Engineering
Entry address(es):
  IP address: 10.0.0.2
  CLNS address: 49.0000.1111.2222.4444.00
Platform: cisco 3640,  Capabilities: Router Switch
Interface: Serial1/0,  Port ID (outgoing port): Serial1/0
Holdtime : 154 sec

Version :
Cisco Internetwork Operating System Software
IOS (tm) 3600 Software (C3640-JK9O3S-M), Version 12.3(22), RELEASE
SOFTWARE (fc2)
Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2007 by cisco Systems, Inc.
Compiled Wed 24-Jan-07 18:02 by ccai

advertisement version: 2
VTP Management Domain: ''
```

Printre alte detalii, comanda poate informa cu privire la adresa IP a vecinului și, de asemenea, poate afișa versiunea sistemului de operare pe care acesta îl rulează.

CDP poate fi activat sau dezactivat atât la nivel global, pentru toate interfețele ruterului, cât și pentru fiecare interfață în parte. La nivel global, comenzile sunt **cdp run** și **no cdp run**, iar la nivel de interfață, comenzile sunt **cdp enable** și **no cdp enable**.

```
Marketing(config)#cdp run
Marketing(config)#int serial1/0
Marketing(config-if)#no cdp enable
Marketing(config-if)#end
Marketing#show cdp interface
Serial1/1 is administratively down, line protocol is down
  Encapsulation HDLC
  Sending CDP packets every 60 seconds
  Holdtime is 180 seconds
Serial1/2 is administratively down, line protocol is down
  Encapsulation HDLC
  Sending CDP packets every 60 seconds
  Holdtime is 180 seconds
Serial1/3 is administratively down, line protocol is down
  Encapsulation HDLC
  Sending CDP packets every 60 seconds
  Holdtime is 180 seconds
```

[1] CDP este unul dintre puținele protocoale care extind semnificația numelui (*hostname*) mai departe de ruterul local.

În exemplul de mai sus, CDP a fost activat la nivel global prin comanda **cdp run** și apoi dezactivat la nivelul interfeței Serial 1/0. În consecință, interfața Serial 1/0 nu mai apare în lista interfețelor comenzii **show cdp interface**, care trimit și primesc mesaje CDP.

Comanda **show cdp** menționează faptul că mesajele CDP se trimit, în mod implicit, la 60 de secunde iar un ruter este eliminat din tabelele CDP ale vecinilor după 180 de secunde (*holdtime*) de inactivitate.

CDP poate fi folosit la „explorarea" rețelelor neadocumentate, deoarece permite administratorului să determine echipamentele și conexiunile dintre ele la fiecare pas.

1.3.2.2 Interfețe de loopback

Interfețele loopback sunt interfețe virtuale, create în software, de către sistemul de operare și nu au nicio componentă hardware asociată. Adresele IP pot fi asignate interfețelor de loopback în aceeași manieră ca și pentru interfețele fizice.

Utilizările interfețelor loopback pot fi numeroase, dar în multe cazuri sunt folosite pentru asignarea unei adrese IP unui ruter în mod global, fără ca această adresă să depindă de o interfață fizică anume. Avantajul constă în faptul că ruterul poate fi adresat după adresa configurată pe o interfață loopback indiferent de care interfețe fizice sunt active la un moment dat. De asemenea, interfețele loopback sunt folosite de unele protocoale de rutare și sunt deseori create în procesele de testare ale rețelelor și protocoalelor.

O interfață de loopback este complet emulată de către sistemul de operare, astfel că va fi activă atâta timp cât ruterul este funcțional , chiar dacă nu are nicio interfață fizică activă. Un ruter poate avea oricâte interfețe de loopback, în limita resurselor sale hardware. Crearea unei interfețe loopback se face în momentul în care este accesat modul de configurare al interfeței:

```
Marketing(config)#interface loopback 0
Marketing(config-if)#ip address 192.168.99.1 255.255.255.252
Marketing(config-if)#exit
Marketing(config)#interface loopback 1
Marketing(config-if)#ip address 172.16.15.1 255.0.0.0
Marketing(config-if)#end
Marketing#
```

Interfețele loopback nu necesită activarea prin comanda **no shutdown**.

1.3.3 Verificarea configurațiilor de bază

În exemplele anterioare s-a arătat faptul că întregul set de comenzi ce descriu configurația din memoria RAM sau NVRAM poate fi vizualizat prin comenzile **show running-config** și **show startup-config**.

O comandă ce sumarizează o mare parte din informațiile ce caracterizează din punct de vedere fizic ruterul curent este **show version**:

```
Engineering#show version
Cisco Internetwork Operating System Software
IOS (tm) C2600 Software (C2600-IK9O3S3-M), Version 12.3(9e), RELEASE
SOFTWARE ()
Copyright (c) 1986-2005 by cisco Systems, Inc.
Compiled Thu 11-Aug-05 22:59 by ssearch
Image text-base: 0x80008098, data-base: 0x819E5F60

ROM: System Bootstrap, Version 11.3(2)XA4, RELEASE SOFTWARE (fc1)

R1 uptime is 5 weeks, 5 days, 10 hours, 42 minutes
```

```
System returned to ROM by reload
System image file is "flash:c2600-ik9o3s3-mz.123-9e.bin"


This product contains cryptographic features and is subject to United
[… linii omise]
cisco 2620 (MPC860) processor (revision 0x102) with 61440K/4096K bytes of
memor.
Processor board ID JAD04480AND (1263225075)
M860 processor: part number 0, mask 49
Bridging software.
X.25 software, Version 3.0.0.
Basic Rate ISDN software, Version 1.1.
1 FastEthernet/IEEE 802.3 interface(s)
2 Low-speed serial(sync/async) network interface(s)
1 ISDN Basic Rate interface(s)
32K bytes of non-volatile configuration memory.
16384K bytes of processor board System flash (Read/Write)

Configuration register is 0x2102
```

În afară de versiunea efectivă a sistemului de operare și platforma pentru care este destinat, comanda **`show version`** informează cu privire la o multitudine de parametri hardware ai ruterului:

- Versiunea programului de *bootstrap* și a *bootloader*-ului;
- Timpul de rulare (*uptime*);
- Ultimul mod de pornire: restartare, pornire simplă, inițializare din ROM, etc;
- Numele fișierului sistemului de operare și locul din care acesta a fost încărcat;
- Modelul procesorului și frecvența procesorului;
- Memoria RAM disponibilă (afișată sub forma X / Y unde X reprezintă memoria disponibilă proceselor și sistemului de operare, Y este memoria pentru buffer-ul de pachte iar suma lor reprezintă cantitatea totală de memorie RAM instalată);
- Numărul de interfețe de rețea instalate, cu tipurile acestora;
- Mărimea memoriei NVRAM;
- Mărimea memoriei *Flash*;
- Valoarea registrului de configurare[1] - singura modalitate de a vizualiza această valoare.
- Aceeași comandă este disponibilă și pe o varietate de alte echipamente Cisco, precum switchuri și firewall-uri.

Deși comanda **`show version`** afișează numărul și tipul tuturor interfețelor ruterului, nu oferă informații despre parametrii configurabili ai acestora. Informații detaliate despre fiecare interfață a unui ruter pot fi obținute prin comenzile **`show interfaces`** și **`show ip interfaces`**. Prima comandă listează cu preponderență parametrii de nivel 2 și statistici la nivel de pachete și octeți iar cea de-a doua afișează parametrii ce privesc configurația IP și diverse protocoale active pe interfața respectivă. Ambele comenzi vor afișa informații despre toate interfețele ruterului și suportă câte o interfață specificată drept parametru pentru a reduce cantitatea de informații afișată.

Exemplu **`show interface serial 1/0`**:

```
Engineering#show interface serial 1/0
Serial1/0 is up, line protocol is up
  Hardware is M4T
  Internet address is 10.0.0.2/24
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
```

[1] Registrul de configurare reprezintă o valoare de 2 octeți ale căror valori, în diferite combinații, desemnează un anumit comportament la pornirea ruterului (metode de inițializare, localizarea sistemului de operare sau a fișierului de configurare, etc). Spre exemplu, valoarea 0x2142(hex) va cauza ignorarea fișierului de configurare din NVRAM. Valoarea uzuală este 0x2102. Practic, el funcționează asemenea unui set de *jumper*-i de pe o placă de bază.

```
     reliability 255/255, txload 1/255, rxload 1/255
  Encapsulation HDLC, crc 16, loopback not set
  Keepalive set (10 sec)
  Restart-Delay is 0 secs
  Last input 00:00:03, output 00:00:03, output hang never
  Last clearing of "show interface" counters never
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
  Queueing strategy: fifo
  Output queue: 0/40 (size/max)
  5 minute input rate 1000 bits/sec, 0 packets/sec
  5 minute output rate 2000 bits/sec, 0 packets/sec
     879 packets input, 644263 bytes, 0 no buffer
     Received 878 broadcasts, 0 runts, 0 giants, 0 throttles
     0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
     939 packets output, 664067 bytes, 0 underruns
     0 output errors, 0 collisions, 2 interface resets
     0 output buffer failures, 0 output buffers swapped out
     3 carrier transitions     DCD=up  DSR=up  DTR=up  RTS=up  CTS=up
```

Exemplu **show ip interface serial 1/0**:

```
Engineering#show ip interface serial1/0
Serial1/0 is up, line protocol is up
  Internet address is 10.0.0.2/24
  Broadcast address is 255.255.255.255
  Address determined by non-volatile memory
  MTU is 1500 bytes
  Helper address is not set
  Directed broadcast forwarding is disabled
  Outgoing access list is not set
  Inbound  access list is not set
  Proxy ARP is enabled
  Local Proxy ARP is disabled
  Security level is default
  Split horizon is enabled
  ICMP redirects are always sent
  ICMP unreachables are always sent
  ICMP mask replies are never sent
  IP fast switching is enabled
```

De regulă este utilă afișarea facilă a stării tuturor interfețelor, cu cei mai importanți parametri, în special pentru procesele de *troubleshooting*. Pentru a lista „pe scurt" stările tuturor interfețelor și adresele configurate, se poate folosi comanda **show ip interface brief**:

```
Engineering#show ip interface brief
Interface          IP-Address      OK? Method Status                Protocol
Serial1/0          10.0.0.2        YES NVRAM  up                    up
Serial1/1          unassigned      YES NVRAM  administratively down down
Serial1/2          unassigned      YES NVRAM  administratively down down
Serial1/           unassigned      YES NVRAM  administratively down down
Loopback0          192.168.2.1     YES NVRAM  up                    up
```

Tabelul returnat listează toate interfețele (inclusiv cele virtuale, precum cea de loopback), adresele IP configurate, modul în care s-a realizat configurarea (NVRAM denotă configurarea statică – o altă valoarea posibilă ar fi fost DHCP), precum și starea interfeței la nivel 1 (*Status*) și starea protocolului de nivel 2 (*Protocol*). La nivel fizic, o interfață ce nu a fost încă activată manual va afișa „*administratively down*" iar o interfață ce nu poate realiza o legătură datorită unei erori va afișa „*down*".

Cisco IOS încorporează și utilitarele „clasice" de testare a conectivității, precum „ping" și „traceroute". Comanda **ping** trimite 5 pachete ICMP Echo și afișează statisticile pachetelor Echo - Reply primite:

```
Router1#ping 12.0.0.2
Type escape sequence to abort.
```

```
Sending 5, 100-byte ICMP Echos to 12.0.0.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/6/8 ms
```

Cele 5 semne de exclamare reprezintă cele 5 pachete Echo-Reply primite cu succes. Simbolurile mai pot cuprinde „.” (punct), ceea ce denotă un timeout sau un „U”, care reprezintă un mesaj „*Unreachable*”.

De asemenea, comanda **`ping`** poate fi introdusă și fără argumentul adresei destinație, pentru a efectua ceea ce se numește un „ping extins”:

```
Router1#ping
Protocol [ip]:
Target IP address: 12.0.0.2
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Source address or interface: 20.0.0.1
Type of service [0]:
Set DF bit in IP header? [no]:
Validate reply data? [no]:
Data pattern [0xABCD]:
Loose, Strict, Record, Timestamp, Verbose[none]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 12.0.0.2, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/7/9 ms
```

Se poate observa că în acest caz pot fi configurați mai mulți parametri, majoritatea fiind opțiuni ale antetului IP.

Comanda **`traceroute`** este folosită pentru a descoperi ruta pe care un pachet o parcurge până la destinație. Pachetele ICMP trimise cu **`traceroute`** primesc succesiv valori din ce în ce mai mari ale câmpului TTL, începând cu valoarea 1 ceea ce cauzează pachetului să „expire” la primul ruter întâlnit. În momentul în care câmpul TTL ajunge la 0, pachetul nu mai este rutați se trimite înapoi spre sursă un pachet ICMP *Time Exceeded*. Celelalte pachete vor „expira” peste un număr tot mai mare de hopuri, până la destinația propriu-zisă.

Exemplu **`traceroute`**:

```
Router1#traceroute 34.0.0.4
Type escape sequence to abort.
Tracing the route to 34.0.0.4
1 12.0.0.2 4 msec 4 msec 4 msec
2 23.0.0.3 20 msec 16 msec 16 msec
3 34.0.0.4 16 msec *  16 msec
```

1.3.4 Facilități ale liniei de comandă

Interfețele în linie de comandă au fost întotdeauna cel mai ușor extensibile și adaptabile. Totuși, o limitare importantă rămâne faptul că interacțiunea în ambele sensuri se face exclusiv în mod text, iar pentru un administrator această limitare se traduce în introducerea dificilă a comenzilor complexeși imposibilitatea memorării mulțimii tutu ror comenzilor disponibile la un moment dat.

Cisco a introdus o serie de facilități care accelerează atât procesul de scriere a comenzilor cât și cel de memorare a sintaxei lor, descrise sumar în cele ce urmează.

O serie de comenzi suportă **parametri impliciți**, în special pentru confirmări sau atunci când opțiunile sunt reduse, ca în exemplul comenzii **reload**, care repornește ruterul și cere o confirmare înainte de a se executa:

```
Engineering#reload
Proceed with reload? [confirm]
```

Opțiunile între paranteze pătrate sunt implicite și pot fi alese doar prin apăsarea tastei Enter. Pentru alte opțiuni este necesară scrierea lor explicită.

Rezultatele unor comenzi pot avea lungimi considerabile, astfel că textul lor se poate extinde pe **pagini multiple**. Prezența textului **--More--** la finalul unei pagini indică faptul că utilizatorul poate apăsa tasta Enter pentru a derula încă o linie sau tasta Space pentru a trece la pagina următoare[1]. De asemenea, este disponibilă și funcția de căutare pentru astfel de rezultate, obținută prin tastarea **/cuvant** în condțiile unui text pe mai multe pagini, fapt ce cauzează derularea automată la prima apariție a lui „cuvant".

Exemplu:

```
Engineering#show running-config
Building configuration...

Current configuration : 1475 bytes
!
version 12.3
no service password-encryption
!
hostname Engineering
[fragment]
 --More--
```

Afișarea comenzilor permise este posibilă în orice moment, chiar și pentru comenzi parțial introduse. Pentru aceasta, se folosește caracterul „?" fără a fi urmat de tasta Enter. Spre exemplu:

```
Engineering(config)#?
Configure commands:
  aaa                             Authentication, Authorization and
Accounting.
  aal2-profile                    Configure AAL2 profile
  access-list                     Add an access list entry
  alarm-interface                 Configure a specific Alarm Interface Card
  alias                           Create command alias
  alps                            Configure Airline Protocol Support
  appletalk                       Appletalk global configuration commands
  arap                            Appletalk Remote Access Protocol
  arp                             Set a static ARP entry
  async-bootp                     Modify system bootp parameters
  atm                             Enable ATM SLM Statistics
  backhaul-session-manager        Configure Backhaul Session Manager
  banner                          Define a login banner
 --More--
```

Apăsarea lui „?" afișează automat toate comenzile disponibile pentru acel context, în ordine alfabetică și urmate de o scurtă descriere (pentru exemplul de mai sus, contextul reprezintă rădăcina modului de configurare). Deoarece majoritatea comenzilor în Cisco IOS sunt formate din mai multe cuvinte/parametri, caracterul „?" va afișa doar variantele pentru următorul cuvânt din comandă. Spre exemplu:

```
Engineering(config)#banner ?
  LINE              c banner-text c, where 'c' is a delimiting character
```

[1] Comportamentul este similar cu cel al utilitarului `more` din Linux.

```
  exec              Set EXEC process creation banner
  incoming          Set incoming terminal line banner
  login             Set login banner
  motd              Set Message of the Day banner
  prompt-timeout    Set Message for login authentication timeout
  slip-ppp          Set Message for SLIP/PPP
```

În exemplul de mai sus, prin comanda **banner ?** se cere listarea tuturor argumentelor ce pot urma comenzii **banner**. Este necesară introducerea unui spațiu înaintea lui „?" pentru a obține lista următorilor parametri. Dacă nu se introduce spațiul respectiv, IOS va încerca să afișeze toate comenzile care încep cu șirul de caractere introdus până la semnul „?", ca în exemplul următor:

```
Engineering#co?
configure  connect  copy
```

Conform rezultatului de mai sus, comenzile **configure**, **connect** și **copy** sunt toate comenzile ce încep cușirul de caractere „co" și pot fi introduse în modul *Privileged EXEC*. Același procedeu poate fi aplicat pentru orice parametru al unei comenzi, nu doar pentru primul.

În unele cazuri, un șir de caractere urmat de semnul „?" va afișa un singur rezultat posibil. Spre exemplu, deducând din rezultatul de mai sus, comanda **conf?** va afișa o singură comandă posibilă, **configure**. Pentru aceste cazuri, IOS va considerașirul de caractere neambiguu ca fiind suficient pentru a identifica o singură comandă și îl va considera corect.

```
Engineering#conf?
configure

Engineering#conf ?
  memory            Configure from NV memory
  network           Configure from a TFTP network host
  overwrite-network Overwrite NV memory from TFTP network host
  terminal          Configure from the terminal
  <cr>

Engineering#conf t?
terminal

Engineering#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Engineering(config)#
```

Exemplul de mai sus demonstrează faptul cășirul **conf t** este șirul minimal ce identifică în mod unic comanda **configure terminal**. Este utilă determinarea scurtăturilor de acest tip pentru comenzile uzuale deoarece scurtează considerabil procesul de configurare. (alte exemple: **en** pentru **enable**, **int** pentru **interface**, **ip add** pentru **ip address**, **no sh** pentru **no shutdown**, etc).

De asemenea, pentru cazurile comenzilor incomplete dar care nu cauzează ambiguități, IOS permite **completarea automată** a comenzii folosind tasta Tab. Spre exemplu,șirul **conf** urmat de tasta Tab se va transforma în **configure**, dar acest lucru nu va influența cu nimic efectul comenzii, tasta Tab oferind doar un feedback vizual și o confirmare a unicității comenzii.

Alte scurtături utile ale liniei de comandă:

- Ctrl-Z întoarce promptul în modul Privileged EXEC;
- Săgețile sus și jos reiau în ordine inversă ultimele comenzi executate;
- Ctrl-R repetă șirul de caractere introdus până în acel moment, util atunci când tastarea unei comenzi e întreruptă de mesajele ruterului;
- Ctrl-A duce cursorul la începutul liniei;
- Ctrl-E duce cursorul la sfârșitul liniei.

Un important mod de a scurta timpul de introducere a comenzilor este **definirea alias-urilor** ce funcționează asemenea unor „abrevieri" ale comenzilor dorite. Un exemplu de definire al unui alias:

```
Router(config)#alias exec s show ip int brief
```

Comanda anterioară creează un alias al comenzii **`show ip interface brief`**, denumit **`s`**, pentru modul EXEC. Din acest moment, comanda va putea fi apelată mult mai simplu, prin tasta „s" urmată de Enter. Pentru a vizualiza *alias*-urile configurate și cele implicite se poate folosi comanda **`show aliases`**.

2 Funcţiile unui ruter

2.1 Rutarea

Prin intermediul interfeţelor de reţea, ruterele interconectează reţele distincte şi stabilesc după diferite criterii ce căi vor urma pachetele pentru a ajunge la destinaţie. În mod concret, un ruter va analiza cel puţin adresa destinaţie a protocolului de nivel 3 (IP) şi va căuta în tabela de rutare o rută care corespunde acestei destinaţii. Ruta c orespunzătoare va specifica fie adresa ruterului următor (*next-hop*), sau o interfaţă de ieşire, dacă ruterul e direct conectat la reţeaua destinaţie. În mod evident, funcţionarea corectă a procesului de rutare este strâns legată de construirea corectă a tabelei de rutare şi introducerea rutelor pentru toate destinaţiile necesare.

Un ruter este considerat un „hop" în calea unui pachet. Un pachet poate parcurge o multitudine de hopuri până la destinaţie, fiecare ruter de pe parcurs luând decizii în mod independent cu privire la calea pe care o va urma pachetul, în continuare. În aceste condiţii ruterele trebuie să aibă informaţii consistente într -o anumită măsură cu privire la rutele cunoscute. În caz contrar, dacă un ruter primeşte un pachet pentru care nu cu noaşte calea spre destinaţie, pachetul nu va putea fi trimis mai departe.

Asigurarea consistenţei rutelor într-o topologie cu mai multe rutere se poate face în două moduri:

- manual, prin configurarea individuală a rutelor (rutare statică);
- automat, folosind protocoale de rutare ce comunică informaţiile despre fiecare reţea cunoscută tuturor celorlalte rutere şi calculează cele mai scurte căi spre acestea.

Totodată, consisterţa informaţiilor de rutare într -o topologie urmăreşte şi scopul evitării buclelor de rutare, adică situaţii în care pachetele vor ciclează între aceleaşi rutere, fără a mai ajunge la destinaţie, datorită unei configuraţii defectuoase.

În cazul configurării manuale a rutelor, administratorul trebuie să se asigure că buclele de rutare nu se pot forma. Pe de altă parte folosirea unui protocol de rutare aduce avantajul mecanismelor de evitare a buclelor construite în fiecare dintre aceste protocoale.

În antetul protocolului IP, câmpul TTL este decrementat cu 1 la fiecare hop pe care pachetul îl traversează, deci de fiecare dată când acesta este rutat. Pachetul nu mai este trimis mai departe când această valoare ajunge la 0. Acest câmp elimină totodată şi situaţia în care un pachet ar cicla la infinit într-o buclă de rutare.

2.1.1 Tabela de rutare

Tabela de rutare reprezintă o structură de date ce stochează informații despre destinațiile cunoscute de un ruter. Practic, dacă nicio rută din tabela de rutare nu poate indica o cale spre o anumită rețea, un ruter nu va putea trimite pachete spre acea rețea (destinația e considerată *unreachable*).

O rută oferă următoarele informații:

- rețeaua pe care o adresează, indicată printr-o adresă de rețea și masca respectivă;
- interfața de ieșire sau adresa *next-hop*-ului ce duce spre destinație;
- originea rutei: direct conectată, configurată static sau învățată printr-un protocol de rutare;
- metrica și distanța administrativă sunt parametri folosiți pentru a distinge între rute multiple spre aceeași destinație.

Originea rutei nu influențează modul de construire a tabelei de rutare.

Comanda de vizualizare a tabelei de rutare este **`show ip route`**:

```
Engineering#sh ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2
       i - IS-IS, su - IS-IS summary, L1 - IS-IS level-1, L2 - IS-IS
level-2
       ia - IS-IS inter area, * - candidate default, U - per-user static
route
       o - ODR, P - periodic downloaded static route

Gateway of last resort is not set

S    221.54.2.0/24 [1/0] via 10.0.0.1
R    192.168.99.0/24 [120/1] via 10.0.0.1, 00:00:03, Serial1/0
     10.0.0.0/24 is subnetted, 1 subnets
C       10.0.0.0 is directly connected, Serial1/0
     199.0.0.0/32 is subnetted, 1 subnets
O       199.0.0.1 [110/65] via 10.0.0.1, 00:14:25, Serial1/0
C    89.0.0.0/8 is directly connected, Loopback1
R    172.0.0.0/8 [120/1] via 10.0.0.1, 00:00:03, Serial1/0
C    192.168.0.0/16 is directly connected, Loopback0
```

În prima parte a tabelei de rutare sunt listate codurile ce desemnează originea rutei iar cea mai mare parte dintre ele indică protocoale de rutare.

Rutele marcate ca fiind *„directly connected"* reprezintă rețelele căror a le aparțin interfețele active ale ruterului. Rețelele direct conectate sunt automat introduse în tabela de rutare, alături de interfețele de ieșire corespunzătoare. În ultimă instanță, rețeaua oricărei destinații accesibile este direct conectată la o interfață a unui ruter iar scopul rutelor și al protocoalelor de rutare este, de fapt, de a găsi drumul spre ruterul care are rețeaua respectivă direct conectată.

Un ruter poate ruta în mod implicit între toate rețelele direct conectate, începând cu momentul în care interfețele respective au adrese IP configurate. Rutele direct conectate persistă în tabela de rutare atâta timp cât interfața membră a rețelei respective este încă activă. Masca de rețea a rutelor direct conectate este masca ce a fost configurată pe interfața în cauză.

În momentul în care un ruter primește un pachet IP pe una dintre interfețe, adresa destinație din antetul pachetului va fi căutată în tabela de rutare, de la cele mai particulare prefixe la cele mai generale. În cazul în care se găsește o rută ce corespunde destinației căutate se consideră două cazuri:

- dacă ruta specifică o interfață de ieșire, pachetul e trimis direct pe interfața respectivă;

- dacă ruta specifică adresa unui *next-hop*, ruterul va mai face o căutare în tabela de rutare pentru a determina interfața de ieșire a rutei direct conectate la rețeaua al cărui membru e *next-hop*-ul respectiv.

Spre exemplu, pentru tabela de rutare anterioară, dacă ruterul primește un pachet cu destinația 221.54.2.40, această destinație va corespunde primei rute (statice), care specifică faptul că pachetele spre această destinație trebuie trimise la *next-hop*-ul 10.0.0.1. Ruterul caută apoi o rețea direct conectată pentru destinația 10.0.0.1 și găsește cea de-a treia rută, spre 10.0.0.0/24, cu interfața de ieșire Serial1/0, pe care va fi trimis și pachetul.

2.1.2 Rutarea statică

S-a menționat anterior faptul că rutele direct conectate ale interfețelor active sunt incluse automat în tabela de rutare iar un ruter poate să ruteze în mod implicit între aceste rețele, fără configurări suplimentare. Această facilitate este, însă, utilă doar pentru rețele aflate la o distanță de un singur hop una de cealaltă.

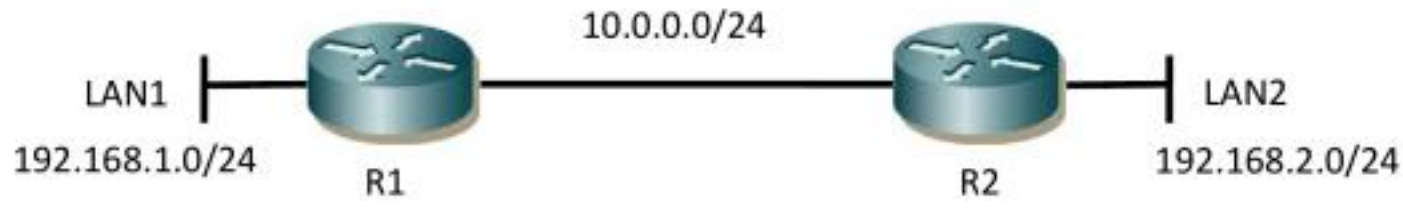

Figura 2-1: Rutarea între rețele direct conectate

Pentru scenariul de mai sus, luând în considerare doar rutele direct conectate, ruterul R1 nu va cunoaște nicio rută spre rețeaua LAN2 din spatele lui R2 deoarece nu își cunoaște decât rutele direct conectate: cea spre LAN1 și spre rețeaua dintre R1 și R2. În acest caz, R1 necesită o rută care să îl indice pe R2 drept *next-hop* pentru destinatia LAN2 (în mod normal este necesară și ruta simetric opusă, de la R2 la LAN1, pentru a avea conectivitate bidirecțională).

Cea mai simplă metodă de a rezolva această situație este definirea unei rute statice spre rețeaua LAN2, ca în exemplul de mai jos:

```
R1(config)#ip route 192.168.2.0 255.255.255.0 10.0.0.2
```

Tabela de rutare a ruterului R1 afișează acum:

```
10.0.0.0/24 is subnetted, 1 subnets
C       10.0.0.0 is directly connected, Serial1/0
C    192.168.1.0/24 is directly connected, Loopback0
S    192.168.2.0/24 [1/0] via 10.0.0.2
```

R1 va putea trimite pachetele cu destinația LAN2 către R2, acesta din urmă identificând rețeaua destinație a pachetelor ca fiind una dintre rețelele sale direct conectate.

Sintaxa comenzii **`ip route`** conține rețeaua spre care se definește ruta, împreună cu masca aferentă. Ultimul parametru poate fi adresa interfeței ruterului vecin, prin care trebuie să treacă traficul, sau direct numele unei interfețe de ieșire. Definirea unei rute statice nu depinde de distanța dintre ruterul curent și destinația în cauză.

Tabela de rutare poate fi folosită și ca filtru de pachete pe baza destinației. Cisco IOS definește interfața virtuală Null0, supranumită și *bit bucket*. Pachetele trimise spre această interfață vor fi

pierdute. Este posibilă crearea unei rute statice cu această destinație pentru pachetele ce trebuie filtrate înainte de a ajunge la destinație:

```
R1(config)#ip route 110.0.0.0 255.0.0.0 null0
R1#show ip route
[...]
S    110.0.0.0/8 is directly connected, Null0
```

O categorie specială de rute statice o reprezintă rutele *default*. O rută *default* este concepută astfel încât să se potrivească oricărei destinații. Există două situații în care se folosesc rutele default:

1. Tabela de rutare este parcursă de la rutele cele mai particulare la cele mai generale, deci de la măștile de rețea cele mai lungi la cele mai scurte; ruta *default* este cea mai generală rută și va fi folosită atunci când destinația unui pachet nu este găsită în niciuna din celelalte rute ale tabelei de rutare.
2. Dacă ruterul are o singură legătură spre un alt ruter este inutilă menținerea mai multor rute, toate cu aceeași destinație de ieșire; pentru un astfel de ruter, o rută *default* este suficientă pentru toate pachetele destinate în afara rețelelor sale direct conectate.

O rută *default* este marcată în tabela de rutare prin simbolul „*". Exemplul următor demonstrează crearea unei rute *default*:

```
R1(config)#ip route 0.0.0.0 0.0.0.0 serial1/0
R1(config)#do show ip route
Codes: C - connected, S - static, R - RIP, [...]

Gateway of last resort is 0.0.0.0 to network 0.0.0.0

     10.0.0.0/24 is subnetted, 1 subnets
C       10.0.0.0 is directly connected, Serial1/0
C    192.168.1.0/24 is directly connected, Loopback0
S    192.168.2.0/24 [1/0] via 10.0.0.2
S*   0.0.0.0/0 is directly connected, Serial1/0
```

O rută de tipul 0.0.0.0 0.0.0.0 este denumită și „ *quad-zero*". Masca /0 permite acestei rute să accepte orice destinație; neavând o zonă de rețea în adresă, masca specifică faptul că toți biții săi pot lua orice valori.

2.1.3 Protocoale de rutare

Rutarea statică poate fi preferată în topologii mici deoarece oferă control deplin și introduce un *overhead* minim din punctul de vedere al procesării. Pe de altă parte, o soluție bazată exclusiv pe rute statice suferă de lipsa scalabilității și de un risc tot mai mare de apariție a erorilor, pe măsură ce rețeaua crește.

Protocoalele de rutare adresează acest neajuns propagând rutele (și cunoscute ca rute direct conectate) în tabelele de rutare ale tuturor ruterelor ce rulează același protocol de rutare. Ruterele schimbă între ele informații de rutare și adaugă automat informațiile necesare la propriile tabele de rutare.

Protocoalele de rutare determină cea mai bună cale spre fiecare rețea destinație utilizând algoritmi proprii. Un avantaj major al multor protocoale de rutare este faptul că informațiile de rutare actualizate sunt schimbate de fiecare dată când în rețea are loc o schimbare. Acest lucru permite ruterelor să învețe dinamic rețelele nou conectate dar și să găsească rute alternative în cazul în care unele legături devin inutilizabile.

În comparație cu rutele definite static, protocoalele de rutare necesită mult mai puțin efort de mentenanță din partea administratorului, dar configurarea lor presupune cunoștințe mai avansate ale fiecărui protocol de rutare folosit. Totuși, protocoalele de rutare sunt mult mai „costisitoare"

pentru resursele de procesor şi memorie ale ruterelor. În consecinţă, multe reţele implementează o schemă hibridă, care combină rutarea statică şi protocoalele de rutare.

Timpul de convergenţă al unei reţele este intervalul în care toate tabelele de rutare ale ruterelor ajung într-o stare de consistenţă, deci când toate ruterele cunosc destinaţiile existente şi rutele corecte spre acestea. În general, o reţea este inutilizabilă sau dificil de controlat cât timp converge, de aceea protocoalele de rutare urmăresc să atingă un timp de convergenţă cât mai mic. Convergenţa ţine şi de topologie, ca întreg, din moment ce fiecare ruter depinde de informaţiile transmise de vecinii săi, darşi de fiecare ruter în parte, deoarece fiecare ruter ce participă într-un protocol de rutare trebuie să calculeze independent căile cele mai bune spre toate destinaţiile cunoscute, urmând să introducă aceste rute în tabela de rutare.

2.1.3.1 Parametrii rutelor

Indiferent de originea unei rute, în tabela de rutare sunt întotdeauna preferate rutele mai „particulare" (cu o mască de reţea cât mai lungă). În cazul în care rute din surse diferite oferă căi spre aceeaşi destinaţie, alegerea celei mai bune rute trebuie făcută şi după alte criterii.

Până în acest moment s-a arătat faptul că rutele pot proveni din trei tipuri diferite de surse: rutele direct conectate, care sunt automat adăugate la tabela de rutare, rutele definite static şi rutele învăţate prin intermediul protocoalelor de rutare. În aceste situaţii apar deseori mai multe rute spre aceeaşi destinaţie, dar cu origini diferite. Spre exemplu, pot exista o rută statică şi o rută a unui protocol de rutare spre aceeaşi destinaţie, două rute de la protocoale de rutare diferite sau chiar două rute învăţate prin acelaşi protocol de rutare care duc, prin căi diferite, spre aceeaşi reţea.

Elementul folosit pentru a distinge între originile rutelor este numit **distanţă administrativă**. Din punctul de vedere al ruterului, distanţa admin istrativă desemnează un grad de „încredere" sau preferinţă asupra anumitor rute, în detrimentul altora. În practică, distanţa administrativă determină o ierarhie bine definită a tuturor modurilor posibile în care o rută poate fi originată.

Câteva dintre cele mai des întâlnite valori ale distanţei administrative sunt următoarele:

Metodă / Protocol	Distanţă administrativă (AD)
Rută direct conectată	0
Rută statică	1
EIGRP sumarizată	5
BGP externă	20
EIGRP internă	90
OSPF	110
IS-IS	115
RIP	120
EIGRP externă	170
BGP internă	200
Origine necunoscută	255

Figura 2-2: Tabelul distanţelor administrative

O valoare mai mică a distanţei administrative este întotdeauna preferată. Dacă, spre exemplu, un protocol de rutare introduce o rută în tabela de rutareşi administratorul configurează ulterior o rută statică spre aceeaşi destinaţie, cea de-a doua o va înlocui pe cea dinamică datorită distanţei administrative.

Problema legăturilor multiple spre aceeași destinație rămâne chiar și după ce s-a luat o decizie pe baza distanței administrative. În rețelele în care porțiuni din topologie formează bucle deseori se întâmplă ca un protocol de rutare să anunțe două sau mai multe rute spre aceeași destinație, care au aceeași distanță administrativă, dar urmează căi diferite până la rețeaua respectivă.

Pentru astfel de situații este nevoie de un procedeu de decizie ce implică în mod direct drumurile pe care pachetele le pot lua până la destinație. **Metrica** reprezintă o metodă de cuantificare a legăturilor dintre rutere. Metricile sunt valori numerice atașate rutelor ce au ca scop diferențierea rutelor provenite de la același protocol de rutare. Fiecare protocol de rutare are propria sa metodă de calculare a metricii, iar calcularea acesteia presupune implicarea unor parametri precum numărul de rutere până la destinație, sau lățimea de bandă a fiecărei legături.

În momentul în care un protocol de rutare determină două rute spre aceeași destinație, dacă cele două rute au metrici diferite, va fi preferată ruta cu metrica cea mai mică. Este posibil ca două rute să aibă chiar și aceeași metrică, chiar dacă urmează trasee diferite. În acest caz, rămâne la latitudinea administratorului și a capabilităților protocolului de rutare dacă r utele multiple vor fi folosite în paralel, într-o schemă de *load balancing*, sau nu.

Distanța administrativă și metrica pot fi vizualizate în tabela de rutare prin comanda **`show ip route`** și sunt notate sub forma [AD/metrică] pentru ficare rută:

```
R 192.168.8.0/24 [120/2] via 192.168.4.1, 00:00:26, Serial0/0/1
```

Exemplul de mai sus este al unei rute învățate prin RIP (R), spre rețeaua 192.168.8.0/24. Ruta are distanța administrativă 120 (implicită pentru protocolul RIP) și metrica 2.

2.1.3.2 Clasificarea protocoalelor de rutare

Un domeniu de rutare, denumit și sistem autonom, cuprinde o topologie de rutere aflate sub o administrație comună. De regulă aceste rutere rulează același protocol de rutare. Internetul este structurat ca o rețea de sisteme autonome, deci protocoalele de rutare se împart în două categorii:

- **IGPs**: Interior Gateway Protocols sunt protocoalele de rutare ce rulează în interiorul unui sistem autonom;
- **EGPs**: Exterior Gateway Protocols sunt protocoalele de rutare ce rulează între sisteme autonome distincte.

IGP-urile pot fi împărțite în alte două categorii, după modul lor de funcționare

- **Protocoale distance-vector**. Rutele sunt anunțate ca vectori caracterizați prin distanță și direcție. Distanța este dată de metrica protocolului respectiv, care poate fi chiar și numărul de hopuri până la destinație. Direcția este dată de interfața de ieșire sau de adresa *next-hop*-ului. Exemple: RIP, RIPv2.
- **Protocoale link-state**. Rutele sunt stocate nu doar din perspectiva ruterului curent, dar fiecare instanță a unui protocol de rutare *link-state* păstrează și o topologie internă a rețelei, pe care o deduce din informațiile schimbate cu vecinii săi. Toate ruterele ce rulează același protocol *link-state* trebuie să cunoască aceeași topologie a rețelei. Fiecare ruter calculează rutele cele mai bune din această topologie, pentru a le introduce în tabela de rutare, în mod independent. Exemple: EIGRP, OSPF, IS-IS.

2.2 RIP

2.2.1 Descrierea protocolului

RIP este cel mai vechi protocol de rutare încă utilizat. Este un protocol de tip distance vector, deci nu păstrează o topologie a întregii reţele în memorie ci doar distanţa şi direcţia spre fiecare destinaţie.

Funcţionarea protocolului este relativ simplă. RIP va trimite întregul conţinut al tabelei de rutare pe toate interfeţele ce participă în procesul de RIP, sub forma unor mesaje de broadcast de nivel 3, la fiecare 30 de secunde. Din moment ce ruterele nu propagă mesajele broadcast, fiecare ruter va emite mesajele de broadcast în mod periodic, către toţi vecinii săi.

Din cele de până acum se deduc o serie de dezavantaje ale lui RIP. În primul rând, overhead-ul pe care protocolul îl impune traficului de reţea prin faptul că fiecare ruter trimite la 30 de secunde toată tabela sa de rutare tuturor vecinilor săi poate deveni extrem de repede inacceptabil pentru reţele cu mai mult de câteva rutere. Pe de altă parte, din moment ce mesajele RIP-ului sunt broadcast, nu există o metodă de a izola acest trafic pentru a nu încărca inutil alte dispozitive din reţea (ca staţiile, spre exemplu) cu procesarea acestor mesaje de broadcast. În plus, broadcast-urile pot prezenta şi un risc de securitate, fiind uşor interceptabile de la orice staţie conectată în reţea.

RIP este, de asemenea, şi singurul protocol exclusiv *classful* disponibil în Cisco IOS. Acest lucru înseamnă că reţelele anunţate prin RIP vor fi anunţate fără masca lor corespunzătoare, şi tuturor acestor reţele, la primirea lor de către un alt ruter, li se va ataşa masca implicită a clasei din care fac parte reţelele respective.

Exemplul de mai jos demonstrează o situaţie simplă în care comunicarea între reţele devine imposibilă datorită acestei limitări.

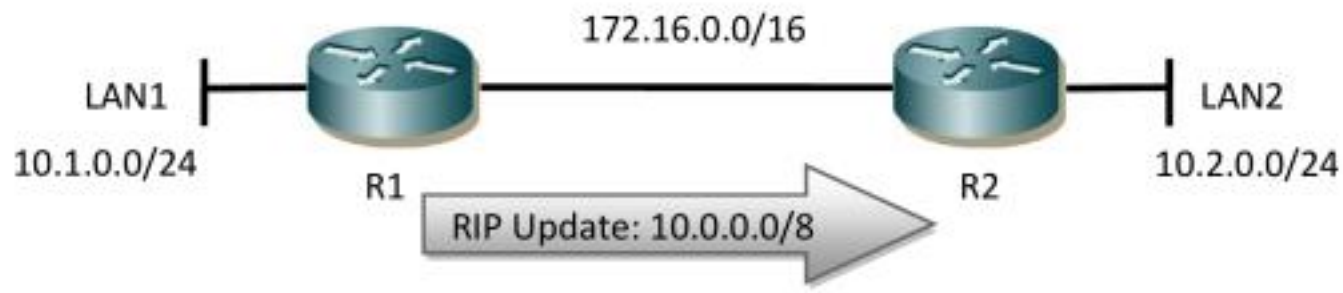

Figura 2-3: Update RIP

În exemplul de mai sus, dacă între cele două rutere se foloseşte RIP, staţiile din reţeaua 10.1.0.0/24 nu vor putea comunica cu cele din reţeaua 10.2.0.0/24. Acest lucru se întâmplă datorită faptului că R2 primeşte prin RIP un update de la R1 pentru reţeaua 10.0.0.0/8 (clasa A) în timp ce R2 are deja reţeaua 10.2.0.0/24 direct conectată, un subnet al reţelei 10.0.0.0/8, cu o mască de reţea mai lungă decât cea primită prin update. Reţeaua sa are un prefix mai lung, deci e mai precisă, şi nu va permite rutei anunţate de RIP să fie stocată în tabela de rutare.

Deoarece măştile de reţea nu sunt transmise în update-urile RIP, diferenţierea dintre reţele se face la nivel *classful*.

Metrica folosită de RIP este cea a *hop-count*-ului. Practic, rutele sunt evaluate din punctul de vedere al metricii în funcţie de numărul de rutere (hopuri) pe care le traversează până la destinaţie. Metrica maximă suportată de RIP este 15, valoarea 16 fiind folosită pentru a desemna o metrică „infinită" sau o rută *unreachable*.

2.2.2 Configurarea protocolului

Înainte de configurarea oricărui protocol de rutare trebuie să existe, în prealabil, o configurație IP a cel puțin unei interfețe active. Cu alte cuvinte, tabela de rutare a unui ruter trebuie să conțină cel puțin rețelele direct conectate.

Minimal, configurarea RIP are două etape:

- intrarea în modul de configurare a protocolului de rutare prin comanda **`router rip`**;
- definirea interfețelor ce participă în protocolul de rutare prin comenzi de tip **`network`**.

```
Router(config)#router ?
  bgp         Border Gateway Protocol (BGP)
  eigrp       Enhanced Interior Gateway Routing Protocol (EIGRP)
  isis        ISO IS-IS
  iso-igrp    IGRP for OSI networks
  mobile      Mobile routes
  odr         On Demand stub Routes
  ospf        Open Shortest Path First (OSPF)
  rip         Routing Information Protocol (RIP)

Router(config)#router rip
Router(config-router)#
```

Ajutorul oferit de linia de comandă poate fi folosit și pentru a lista toate protocoalele de rutare suportate de ruter, ca în exemplul de mai sus. Se observă că intrarea în modul de configurare al unui protocol de rutare este reflectată și in modul de configurare al prompt-ului: **`config-router`**. Comanda nu pornește efectiv procesul RIP, ci doar creează contextul pentru comenzile de configurare ale acestuia. Închiderea unui protocol de rutare se face prin așezarea cuvântului **`no`** înainte de comanda **`router`** și numele protocolului dorit. Ex: **`no router rip`**. Închiderea unui protocol de rutare are ca efect și eliminarea configurațiilor acestuia.

Înainte ca protocolul să poată efectiv funcționa, ruterul trebuie să știe ce interfețe vor fi folosite pentru a comunica cu ruterele vecine și ce rețele vor fi anunțate în protocolul de rutare. Comanda **`network`** specifică aceste informații, urmată de **adresa classful a uneia sau mai multor rețele direct conectate**.

Exemplu:

```
R1(config)#router rip
Router(config-router)#do show ip int brief
Interface              IP-Address      OK? Method Status                Protocol
Serial1/0              20.0.0.1        YES manual up                    up
Loopback0              100.0.0.1       YES manual up                    up
Loopback1              110.0.0.1       YES manual up                    up
Loopback2              120.0.0.1       YES manual up                    up
R1(config-router)#network 20.0.0.0
R1(config-router)#network 100.0.0.0
R1(config-router)#network 110.0.0.0
R1(config-router)#network 120.0.0.0
R1(config-router)#
```

Comand **`network`** are două efecte distincte:

- activează RIP pe toate interfețele ce se încadrează în adresa de rețea specificată ca parametru pentru a putea primi și trimite update-uri RIP;
- anunță rețelele respectivelor interfețe în update-urile RIP.

Introducerea unei adrese de subrețea va avea ca efect trunchierea adresei la lungimea prefixului clasei respective. Cu alte cuvinte, comanda **`network 192.168.12.64`** introdusă pentru o interfață din rețeaua 192.168.12.64/26 va lua în considerare doar adresa 192.168.12.0, cu masca sa classful (/24).

Activarea protocolului RIPși pe R2, împreună cu adăugarea rețelei 20.0.0.0 prin comanda **network**, va afișa următoarea tabelă de rutare:

```
     100.0.0.0/24 is subnetted, 1 subnets
R       100.0.0.0 [120/1] via 20.0.0.1, 00:00:26, Serial1/0
C    200.0.0.0/24 is directly connected, Loopback0
     20.0.0.0/24 is subnetted, 1 subnets
C       20.0.0.0 is directly connected, Serial1/0
     110.0.0.0/24 is subnetted, 1 subnets
R       110.0.0.0 [120/1] via 20.0.0.1, 00:00:23, Serial1/0
     120.0.0.0/24 is subnetted, 1 subnets
R       120.0.0.0 [120/1] via 20.0.0.1, 00:00:21, Serial1/0
```

Se observă că rutele spre rețelele direct conectate ale lui R1 au fost propagate pe R2 cu metrica 1 (un hop).

Pentru vizualizarea informațiilor generale despre toate protocoalele de rutare care rulează la un moment dat, există comanda **show ip protocols**. Comanda afișează interfețele active în fiecare protocol de rutare, comenzile **network** introduse și vecinii cu care se realizează schimbul de informații.

```
R1#show ip protocols
Routing Protocol is "rip"
  Sending updates every 30 seconds, next due in 24 seconds
  Invalid after 180 seconds, hold down 180, flushed after 240
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Redistributing: rip
  Default version control: send version 2, receive version 2
    Interface             Send  Recv  Triggered RIP  Key-chain
    Serial1/0             2     2
    Loopback0             2     2
    Loopback1             2     2
    Loopback2             2     2
  Automatic network summarization is not in effect
  Maximum path: 4
  Routing for Networks:
    10.0.0.0
    20.0.0.0
    30.0.0.0
    100.0.0.0
    110.0.0.0
    120.0.0.0
  Routing Information Sources:
    Gateway         Distance      Last Update
    Gateway         Distance      Last Update
    20.0.0.2              120      02:00:07
  Distance: (default is 120)
```

În unele situații, precum cele în care o rețea locală formată din stații trebuie anunțată în protocolul de rutare, nu este necesară trimiterea update-urilor de rutare pe acea interfață deoarece nu mai există rutere care să ruleze RIP în rețeaua respectivă. În general, este de dorit evitarea traficului inutil în aceste segmente atât pentru că el poate reprezenta un risc de securitate, putând fi interceptat de către stații, cât și pentru că stațiile vor trebui să proceseze broadcast-urile de nivel 3 până la nivelul transport, pentru a realiza că mesajele respective nu le sunt adresate.

Comanda **passive-interface** adresează aceste situații prin împiedicarea trimiterii broadcast-urilor cu informații de rutare pe interfețele dorite. Chiar daca o interfața este „pasivizată" în acest fel, ea poate să fie inclusă în continuare în update-urile de rutare.

Exemplu:

```
R1(config)#router rip
R1(config-router)#passive-interface serial1/0
```

Ruterele configurate cu o rută default vor folosi această rută de fiecare dată când parcurgerea tabelei de rutare pentru destinația unui pachet nu va returna nicio altă rută. În acest caz, pachetul va fi trimis spre *next-hop*-ul sau interfața de ieșire indicată de ruta *default*.

Este destul de probabil ca un sistem autonom care rulează RIP să aibă o singură conexiune cu exteriorul, spre Internet, de exemplu. Pentru ca toate rutere să poată trimite pachete spre Internet prin această conexiune, ele vor trebui configurate individual cu câte o rută default. RIP, ca și alte protocoale de rutare, oferă posibilitatea propagării unei astfel de rute default alături de celelalte rute ale domeniului de rutare. Avantajul constă într-o administrare mult mai facilă, și în faptul că ruta respectivă va deveni rută default în mod automat pe toate ruterele pe care va fi instalată.

Pentru ca o rută default să poată fi propagată într-un domeniu de rutare RIP, un ruter ce are deja definită o rută default trebuie configurat cu următoarea comandă:

```
R1(config-router)#default-information originate
```

Ruterul va începe să se anunțe pe sine ca fiind *next-hop*-ul rutei default și va fi folosit de către toți vecinii ca destinație pentru pachetele destinate în afara rețelei. Pentru a propaga o rută *default* chiar și în absența unei rute *default* din tabela de rutare se poate adăuga parametrul **always** comenzii de mai sus.

2.2.3 Caracteristici ale protocoalelor distance-vector

Termenul „*distance-vector*" desemnează faptul că ruterele ce rulează astfel de protocoale reține doar o imagine locală a topologiei, în care fiecare destinație este caracterizată printr-o distanță (metrică) și o direcție (interfață de ieșire sau adresă *next-hop*).

În general, protocoalele *distance-vector* trimit întreaga tabelă de rutare la intervale periodice, tuturor vecinilor, folosind mesaje broadcast. La primirea informațiilor de rutare de la vecini, un ruter incrementează metrica rutelor primite și le instalează pe cele necunoscute în propria tabelă de rutare, împreună cu adresa *next-hop* a ruterului de la care au fost primite. Protocoalele de rutare *distance-vector* folosesc algoritmul Bellman-Ford pentru calcularea căilor celor mai scurte către destinațiile cunoscute.

Avantajele protocoalelor *distance-vector* constau în ușurința de administrare și configurare și consumul redus de resurse. În mod evident, orice protocol de rutare folosește mai multe resurse decât o implementare bazată exclusiv pe rutare statică, dar protocoalele *distance-vector* utilizează mai puțină memorie și putere de procesare decât cele *link-state*.

În același timp, unul dintre cele mai mari dezavantaje ale acestor protocoale este convergența greoaie, deoarece timpul de convergență depinde de update-urile periodice. O consecință a acestei limitări este și scalabilitatea redusă, deoarece pe măsură ce o rețea crește, durata de propagare a unei modificări, deci a convergenței, crește. Problema convergenței este una importantă deoarece în timpul stabilirii convergenței unui protocol, tabelele de rutare pot conține informații inconsistente ce pot duce la stabilirea unor bucle rutare.

O soluție pentru accelerarea convergenței este folosirea de *triggered updates*, adică mesaje de update trimise nu doar periodic, după un anumit interval, ci și la apariția unui anumit eveniment ce indică schimbări în topologia rețelei. Chiar și cu această facilitate, timpul de convergență al acestor protocoale este situat mult peste cel al protocoalelor *link-state*.

2.2.3.1 Holddown timers

Deşi folosirea de *triggered updates* accelerează convergenţa, acestea pot cauza efecte dăunătoare în anumite condiţii. O astfel de situaţie este reprezentată de interfeţele instabile (*flapping*), a căror legătură oscilează între *up* şi *down* fie din cauza unei erori de nivel 2 fie din cauza unei imperfecţiuni de proiectare sau instalare la nivelul fizic. Reacţia prea rapidă a unui ruter la fiecare astfel de schimbare a stării unei interfeţe poate cauza bucle de rutare din cauza timpului de propagare a modificării.

Ruterele folosesc un timp de aşteptare (*holddown timer*) pentru a evita reinstaurarea imediată a unei rute picate, în urma unui update de rutare sosit de la un vecin ce poate să nu fi aflat despre modificarea survenită.

Funcţionarea *holddown timers* poate fi redusă la următoarele etape:

1. o reţea este declarată inaccesibilă printr-un mesaj al unui vecin, iar ruterul ce primeşte informaţia consideră ruta ca fiind posibil nedisponibilă şi porneşte *holddown timer*-ul;
2. ruterul ignoră toate update-urile despre acea rută cu metrici egale sau mai defavorabile în perioada *holddown timer*-ului;
3. ruterul acceptă reinstaurarea rutei doar dacă primeşte o rută cu o metrică mai bună decât cea veche; o metrică mai bună asigură faptul că vecinii ruterului au aflat despre modificare şi au calculat alte căi spre destinaţie, deci nu există riscul unei bucle.

O rută marcată ca fiind posibil inaccesibilă va fi folosită pentru a trimite pachete deoarece situaţia poate fi cea prezentată anterior, a unei interfeţe instabile.

2.2.3.2 Split horizon

Split horizon reprezintă o altă tehnică de evitare a buclelor de rutare. Regula *split horizon* enunţă faptul că un ruter nu trebuie să trimită informaţii despre o reţea pe aceeaşi interfaţă prin care a aflat despre acea reţea.

Logica din spatele acestei reguli este uşor de dedus. Atâta timp cât un ruter află pentru prima oară despre o anumită reţea de la unul dintre vecini, se poate considera că vecinul respectiv este mai aproape de acea reţea şi va afla despre eventualele modificări ale sale mai repede decât ruterul care a aflat despre ea din update-urile vecinului. În consecinţă, primul ruter nu are dreptul să trimită informaţii despre aceste rute vecinului său, evitând riscul de a-i suprascrie tabela de rutare cu informaţii neactualizate.

2.2.3.3 Route poisoning

În cazul în care un ruter detectează că o reţea direct conectată nu mai este disponibilă, nu o va mai introduce în update-urile de rutare. Vecinii săi vor păstra ruta pe durata *holddown timer*-ului, până când aceasta va expira din tabela de rutare din lipsa update-urilor. Acest proces poate fi accelerat prin tehnica *route poisoning*.

Folosind *route poisoning*, un ruter ce detectează prezenţa unei reţele nevalide va transmite ruta către acea reţea în update-urile sale cu valoarea metricii 16, ceea ce denotă în RIP o metrică infinită, adică o reţea *unreachable*. Ruta astfel primită de către vecini va fi, de asemenea, trimisă mai departe cu metrica 16, astfel propagându-se în întreaga reţea informaţia despre reţeaua devenită inaccesibilă, nemaifiind nevoie să se aştepte expirarea unor timere.

2.2.4 RIPv2

RIPv2 este definit în RFC1723. Principala îmbunătăţire adusă de RIPv2 este eliminarea funcţionalităţii exclusiv classful. RIPv2 poate să transporte rutele împreună cuştiinţe de reţea

corespunzătoare deși, în mod implicit, sumarizarea la granițele *classful* este încă activă. În plus, RIPv2 eficientizează utilizarea traficului în rețea prin folosirea adresării multicast în locul celei broadcast, spre adresa 224.0.0.9.

Configurarea RIPv2 este identică cu cea pentru RIP. Comanda **network** are aceeași semnificație ca și în prima versiune, dar rutele direct conectate vor fi anunțate folosindu-se măștile de rețea de pe interfețele respective. Singura diferență constă în specificarea versiunii sub modul de configurare al protocolului de rutare:

```
R2(config)#router rip
R2(config-router)#version 2
```

Comanda **show ip protocols** arată versiunea update-urilor trimise și primite (câmpurile *Send* și *Recv*):

```
R2#show ip protocols
Routing Protocol is "rip"
  Sending updates every 30 seconds, next due in 8 seconds
  Invalid after 180 seconds, hold down 180, flushed after 240
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Redistributing: rip
  Default version control: send version 2, receive version 2
    Interface             Send  Recv  Triggered RIP  Key-chain
    Serial1/0             2     2
[...]
```

Un comportament interesant al celor două versiuni ale protocolului RIP este faptul că versiunea 1 este *forward-compatible* cu versiunea 2. Cu alte cuvinte, un ruter care rulează versiunea 1 va trimite doar update-uri *classful*, dar va putea primi update-uri *classless*, de la un ruter care rulează versiunea 2 (deși le va interpreta tot *classful*). Acest lucru este posibil datorită structurii mesajelor de update care permit ignorarea câmpurilor ce conțin prefixul rețelei pentru un ruter ce rulează versiunea 1.

Sumarizarea este activă în mod implicit si pe versiunea 2 a protocolului RIP. Mai exact, dacă un ruter are interfețe conectate la subrețele ale aceleiași rețele *classful*, RIPv2 va propaga în continuare doar rețeaua *classful*, considerând-o o sumarizare a celorlalte rețele. Dezactivarea sumarizării se face prin comanda no **auto-summary**:

```
R2(config)#router rip
R2(config-router)#no auto-summary
```

Posibilitatea sumarizării rutelor este o facilitate deseori utilă pentru că ajută la reducerea tabelelor de rutare și accelerarea procesului de decizie al ruterelor. Din această cauză RIPv2 permite și configurarea sumarizării manuale, în condițiile în care cea automată este dezactivată. Comanda pentru sumarizare se introduce pe interfața care va trimite ruta sumarizată iar sintaxa ei este:

```
R1(config-if)#ip summary-address rip 96.0.0.0 255.0.0.0
```

Adresa și masca reprezintă ruta sumarizată. RIP nu permite decât sumarizarea în adrese de rețea cu masca cel puțin egală cu masca implicită a clasei rețelei sumarizate.

2.3 Liste de acces (ACLs)

O listă de acces (ACL – *Access Control List*) reprezintă un set de instrucțiuni ce descriu parametrii după care sunt selectate anumite tipuri de trafic în vederea filtrării, inspectării sau procesării într-un mod diferit. Una dintre principalele utilizări ale listelor de acces, ce va fi descrisă

în continuare, este aplicarea lor pe interfeţele de reţea ale unui ruter cu scopul de a putea specifica care tipuri de trafic sunt permise prin interfeţele respective şi care nu.

Aplicarea ACL-urilor pe interfeţe cu scopul de a filtra traficul se face conform regulii: **un ACL pentru fiecare protocol al fiecărei direcţii a fiecărei interfeţe**. Deci un ruter cu două interfeţe, care rulează IP, IPX şi AppleTalk poate suporta 2 interfeţe x 2 direcţii x 3 protocoale = 12 ACL-uri.

Un ruter nu are, în mod implicit, niciun ACL configurat, iar traficul este permis necondiţionat atât la intrare cât şi la ieşire, din toate interfeţele. Prezenţa unui ACL la intrarea unei interfeţe, spre exemplu, are ca efect filtrarea traficului înainte ca pachetele acestuia să ajungă în procesul de rutare sau să fie supuse altor prelucrări. Pachetele care trec cu succes de ACL vor fi singurele ce vor fi preluate de ruter.

2.3.1 Modul de funcţionare a listelor de acces

Un ACL conţine o serie de teste care analizează antetele de nivel 3 şi 4 ale pachetelor. Fiecare test este urmat de o decizie binară: *permit* sau *deny*.

Când un ACL este aplicat pe o interfaţă, toate pachetele care circulă în sensul respectiv vor fi confruntate cu testele din ACL. Parcurgerea elementelor unui ACL se face secvenţial şi se opreşte în momentul în care se determină că traficul se încadrează într-unul dintre teste, moment în care se aplică acţiunea corespunzătoare testului respectiv. Dacă acţiunea este *permit*, pachetul este lăsat să treacă la fel ca şi în lipsa unui ACL. Dacă acţiunea este *deny*, pachetul este ignorat fără a se trimite vreun fel de notificări.

Pentru cazul în care un pachet nu se încadrează în condiţiile niciunui test din ACL, pachetul este ignorat, ca şi în cazul unei acţiuni explicite de *deny*. ACL-urile au o acţiune implicită de *deny* la sfârşitul lor, ce nu poate fi eliminată şi care se aplică tuturor pachetelor ce nu sunt acoperite de condiţiile ACL-ului. Din acest motiv aplicarea unui ACL vid are ca efect blocarea completă a traficului şi, totodată, un ACL care nu conţine nicio condiţie care să permită traficul este echivalent cu un ACL vid care blochează orice pachet.

Fiecare test din cadrul unui ACL este executat în mod independent şi poate conţine criterii legate de parametrii de nivel 3 sau 4 ai pachetelor. Spre exemplu, un ACL aplicat pe o interfaţă a unui ruter conectată la o reţea locală, pe direcţia „*in*", poate fi „citit" în felul următor:

- blochează traficul TCP al tuturor staţiilor din LAN către portul 80 al oricărei destinaţii = reţeaua locală nu va mai avea acces la servere web;
- blochează tot traficul ICMP al staţiei X spre serverul Y = filtrarea se face pe baza protocolului ICMP;
- blochează traficul IP (TCP+UDP) al tuturor staţiilor mai puţin cea a şefului de departament la serverele altor departamente în intervalul orar 09-17 = ACL-ul este activ într-un anumit interval orar;
- permite tot traficul.

Ultima condiţie este necesară din cauza instrucţiunii implicite de *deny* de la sfârşitul ACL-ului. Lipsa ultimei instrucţiuni are ca efect crearea unui ACL creat doar din instrucţiuni de *deny*, ceea ce este echivalent cu blocarea completă a traficului din interior spre exterior.

Un exemplu al unui ACL aplicat pe interfaţa dinspre exterior, pentru a proteja accesul la reţeaua locală, poate fi următorul:

- permite traficul din exterior spre LAN doar pentru conexiuni deja stabilite din interior = filtrarea analizează procesul de *handshake* al TCP.

Un ACL care conţine doar această instrucţiune va bloca tot traficul din exterior spre reţeaua locală, mai puţin traficul conexiunilor TCP iniţiate din interior.

În mod evident, ordinea instrucțiunilor dintr-un ACL este extrem de importantă. Spre exemplu, în primul ACL, prezența lui „permite tot traficul" la începutul ACL -ului ar fi făcut toate celelalte instrucțiuni inutile deoarece decizia s-ar fi aplicat din start tuturor pachetelor.

De asemenea, un ACL trebuie proiectați având în vedere scopul său. Dacă traficul este în preponderență permis, cu mici excepții, ACL-ul va fi construit din statement-uri de *deny* urmate de un statement care permite tot restul traficului. În mod similar, dacă traficul este permis doar pentru câteva excepții, statement-urile vor acoperi aceste excepții iar restul traficului va fi ignorat de către statement-ul implicit al fiecărui ACL.

Din moment ce un ACL este parcurs pentru fiecare pachet în parte, se recomandă și rearanjarea intrărilor acestuia în funcție de frecvența apariției anumitor tipuri de trafic, pentru îmbunătățirea performanței.

2.3.2 Tipuri de liste de acces

După structura lor, ACL-urile se împart în două categorii:

- **ACL-uri standard**, care pot filtra traficul doar pe baza informației din câmpul de adresă sursă al pachetelor;
- **ACL-uri extinse**, care pot filtra traficul pe baza tuturor parametrilor de adresare de nivel 3 sau 4.

La creare, ACL-urile primesc un număr de identificare folosit atât pentru a distinge între mai multe ACL-uri configurate pe același ruter cât și pentru a stabili ce tip de ACL se creează:

- numerele în intervalele [1, 99] și [1300, 1999] sunt ale ACL-urilor standard;
- numerele în intervalele [100, 199] și [2000, 2699] sunt ale ACL-urilor extinse.

Intervalele neasignate aparțin altor protocoale de nivel 3. Intervalele de mai sus se referă doar la protocolul IP.

În principiu, un ACL poate fi plasat oriunde în calea unui pachet, de la sursă până la destinație. Totuși, plasarea unui ACL trebuie făcută în funcție de tipul său:

ACL-urile standard trebuie plasate **cât mai aproape destinației** deoarece ele nu specifică o destinație anume, iar plasarea lor aproape de sursă poate să blocheze traficul și spre alte destinații decât cele intenționate.

ACL-urile extinse trebuie plasate **cât mai aproape de sursă** deoarece ele cuprind atât informațiile despre sursă cât și despre destinație iar filtrarea pachetelor mai târziu, pe parcurs, ar crea trafic inutil în rețea.

Conceptele de sursă și destinația din cadrul ACL -urilor standard și extinse au sens doar în momentul în care ACL-ul este aplicat unei interfețe și i se specifică sensul (*in* sau *out*).

2.3.3 Configurarea listelor de acces standard

ACL-urile standard filtrează doar pe baza sursei, deci acesta va fi singurul criteriu de decizie. Sursa pachetelor se specifică sub forma unei adrese și a unui **wildcard mask**.

Pentru adrese de rețea, un *wildcard mask* este calculat ca fiind inversul binar al măștii de rețea. Pentru clarificare, iată câteva exemple de corespondențe între *network mask* și *wildcard mask*:

- mască 255.255.0.0 -> *wildcard* 0.0.255.255
- mască 255.255.255.192 -> *wildcard* 0.0.0.63
- mască 255.252.0.0 -> *wildcard* 0.3.255.255

Într-o astfel de comparație, *wildcard mask*-urile pot fi considerate echivalente (ca utilitate) cu măștile de rețea. În realitate, însă, excepția intervine în faptul că un wildcard *mask* poate conține șiruri discontinue de biți de 1, în timp ce acest lucru nu este permis în cazul măștilor de rețea.

Spre exemplu, pentru crearea unui ACL standard cu identificatorul 10, care permite accesul tuturor stațiilor din rețeaua 192.168.2.0/24 spre orice destinație, se folosește următoarea comandă:

```
R2(config)#access-list 10 permit 192.168.2.0 0.0.0.255
```

Crearea ACL-ului are loc odată cu primul statement introdus. ACL-ul poate fi continuat. Spre exemplu, următoarele două comenzi permit accesul tuturor stațiilor și din rețeaua 172.19.0.0/16 către orice destinație, mai puțin pentru stația 172.19.12.10:

```
R2(config)#access-list 10 deny host 172.19.12.10
R2(config)#access-list 10 permit 172.19.0.0 0.0.255.255
```

Comenzile suplimentare vor fi adăugate la ACL-ul precedent, cu numărul 10, în ordinea în care au fost introduse.

Cuvântul *host* poate fi folosit pentru a desemna adresa unei singure stații, pentru care nu este nevoie de un *wildcard mask*. Următoarele *statement*-uri sunt, deci, echivalente:

```
R2(config)#access-list 10 deny host 172.19.12.10
R2(config)#access-list 10 deny 172.19.12.10 0.0.0.0
```

De asemenea, cuvântul *any* poate fi folosit pentru a desemna orice sursă (sau destinație, pentru ACL-urile extinse). Următoarele *statement*-uri sunt echivalente:

```
R2(config)#access-list 10 permit any
R2(config)#access-list 10 permit 0.0.0.0 255.255.255.255
```

Vizualizarea ACL-urilor configurate poate fi realizată prin comanda **`show access-lists`**. Aceeași comandă afișează și numărul pachetelor ce au făcut *match* pe una dintre condițiile fiecărui ACL, facilitând astfel determinarea ACL-urilor greșit configurate sau aplicate.

Aplicarea unui ACL standard pe o interfață se execută prin comanda la nivel de interfață **`ip access-group`** urmată de numărul ACL-ului de aplicat și direcția sa. Exemplu:

```
R2(config-if)#ip access-group 10 in
```

Din momentul aplicării toate pachetele vor fi supuse testelor din ACL-ul respectiv.

ACL-urile standard pot fi aplicate și terminalelor virtuale ale ruterului (de obicei, acest lucru se realizează doar în direcția „ *in*" pentru a determina ce adrese se pot conecta la aceste terminale). Pentru aplicarea unui ACL unuia sau mai multor terminale virtuale se folosește comanda **`access-class`**, ca în exemplul următor:

```
R2(config)#line vty 0 4
R2(config-line)#acces
R2(config-line)#access-class 10 in
```

Atenție, *statement*-urile dintr-un ACL standard nu pot fi intercalate, deci dacă se dorește inserarea unui *statement* în mijlocul unui ACL sau editarea unuia, lista va trebui ștearsă și refăcută în ordinea corectă. E recomandabil ca ACL-urile să fie create separat, într-un editor de texte, până la definitivarea lor.

2.3.4 Configurarea listelor de acces extinse

În plus, pe lângă verificarea adresei sursă a pachetului, ACL-urile extinse introduc posibilitatea de a verifica adresa destinație a pachetului, precum și protocolul și numerele de port sursă și destinație de la nivelul transport.

Simplificată, sintaxa ACL-urilor extinse este:

access-list [numar] deny/permit [protocol] [informatii_sursa] [informatii_destinatie]

unde:

- [numar] reprezintă numărul ACL-ului și trebuie să fie din intervalul ACL-urilor extinse;
- [protocol] poate cuprinde, printre altele, ip, tcp, udp, icmp, protocoale de rutare și altele. Prin IP se consideră atât TCP cât și UDP;
- [informatii_sursa] și [informații_destinație] pot avea valorile *any*, *host* [adresă], adresă și/sau operatori pentru porturi, servicii și *flag*-uri;
    - operatorii pentru porturi sunt disponibili doar pentru protocoalele TCP sau UDP și includ, printre altele, eq (*equal*), gt (*greater than*), lt (*less than*), neq (*not equal*), *range*, etc, urmați de numele unui serviciu sau numărul unui port;
    - *flag*-urile se referă la câmpuri ale antetelor de nivel 3 sau 4 și includ: rst, ack, established, syn, tos, etc.

Un exemplu de ACL extins este următorul:

```
Router(config)#access-list 101 deny tcp host 192.168.2.1 any eq telnet
Router(config)#access-list 101 permit tcp 192.168.2.0 0.0.0.255 any eq www
Router(config)#access-list 101 deny icmp any host 10.0.0.1 echo
```

ACL-ul de mai sus conține trei *statement*-uri, după cum urmează:

- primul împiedică accesul stației 192.168.2.1 la portul 23 al oricărui server;
- al doilea *statement* permite toate conexiunile TCP inițiate din rețeaua 192.168.2.0/24 către orice server web;
- al treilea blochează pachetele de tip ICMP Echo de la orice stație către stația 10.0.0.1.

Aplicarea ACL-urilor extinse pe interfețe sau terminale virtuale se face la fel ca și în cazul ACL-urilor standard, prin comenzile **ip access-group** și **access-class**.

Comanda **show access-lists** afișează intrările din lista 101, împreună cu numerele de secvență corespunzătoare:

```
Extended IP access list 101
    10 deny tcp host 192.168.2.1 any eq telnet
    20 permit tcp 192.168.2.0 0.0.0.255 any eq www
    30 deny icmp any host 10.0.0.1 echo
```

2.3.5 Configurarea listelor de acces cu nume

ACL-urile au o multitudine de întrebuințări ce nu se reduc doar la filtrarea traficului pe interfețele ruterelor. În momentul în care ACL-urile devin diverse și numeroase în configurația unui ruter, evidența bazată pe numerele devine dificil de urmărit, în special pentru administratorii ce nu au creat configurația respectivă.

Named ACL-urile înlocuiesc identificatorul numeric cu un nume dat de administrator, ce poate conține în câteva cuvinte scopul ACL-ului respectiv. Mai mult decât atât, ACL-urile cu nume suportă inserarea de *statement*-uri în orice poziție liberă fără a mai fi necesară rescrierea întregii liste. ACL-urile cu nume pot fi atât *standard* cât și *extended*.

Spre exemplu, următorul exemplu arată crearea unui named ACL *standard*:

```
R2(config)#ip access-list standard DENY_SERVERS
R2(config-std-nacl)#deny host 192.168.100.1
R2(config-std-nacl)#deny host 192.168.100.15
R2(config-std-nacl)#permit any
```

ACL-ul nou format afișat de comanda **show access-lists** apare în felul următor:

```
Standard IP access list DENY_SERVERS
    10 deny   192.168.100.1
    20 deny   192.168.100.15
    30 permit any
```

În acest moment, dacă se dorește restricționarea traficului pentru o nouă stație, ACL -ul ar trebui rescris din moment ce adăugarea unei noi reguli de *deny* s-ar face după regula *permit any*, și nu ar mai avea efect. Introducerea unui nou *statement* se realizează prin specificarea unui număr liber de secvență pe care acesta să fie introdus:

```
R2(config)#ip access-list standard DENY_SERVERS
R2(config-std-nacl)#7 deny host 192.168.100.99
```

Afișarea ACL-ului demonstrează poziționarea corectă a noului *statement*:

```
Standard IP access list DENY_SERVERS
    7 deny   192.168.100.99
    10 deny   192.168.100.1
    20 deny   192.168.100.15
    30 permit any
```

În mod complet similar se creează și ACL-urile *extended* cu nume. Spre exemplu:

```
R2(config)#ip access-list extended PERMIT_WELL_KNOWN
R2(config-ext-nacl)#permit tcp 192.168.100.0 0.0.0.255 any range 1 1024
R2(config-ext-nacl)#deny ip any any
```

Aplicarea named ACL-urilor pe interfețe se face în aceeași manieră ca și la ACL -urile numerotate, cu diferența că în locul numărului se folosește numele.

2.3.6 Ordonarea și documentarea listelor de acces

În special în cazul named ACL-urilor, inserarea continuă de *statement*-uri în pozițiile libere din numerele de secvență poate duce la un moment dat la imposibilitatea adăugării de noi intrări. Pentru aceste cazuri dar și pentru reorganizarea din punctul de vedere al lizibilității ACL-urilor, Cisco a introdus o facilitate de reordonare a numerelor de secvență pentru ACL-uri.

Reordonarea funcționează pentru orice tip de ACL, *standard*, *extended* sau *named*.

Spre exemplu, pentru a reordona ACL-ul DENY_SERVERS creat anterior începând de la numărul de secvență 20 cu increment de 5, se folosește comanda următoare:

```
R2(config)#ip access-list resequence DENY_SERVERS 20 5
R2(config)#do show access-lists DENY_SERVERS
Standard IP access list DENY_SERVERS
    20 deny   192.168.100.15
    25 deny   192.168.100.1
    30 deny   192.168.100.99
    35 permit any
```

De asemenea, este deseori utilă documentarea funțiilor ACL -urilor în mai mult detaliu decât numele folosit pentru ACL. Facilitatea **remark** permite introducerea de comentarii într-un ACL:

```
R2(config)#ip access-list standard DENY_SERVERS
R2(config-std-nacl)#remark ?
  LINE  Comment up to 100 characters
  <cr>

R2(config-std-nacl)#remark Blocheaza accesul serverelor S1, S15 si S99
```

În *running-config*, comentariul apare ca un statement al ACL-ului:

```
ip access-list standard DENY_SERVERS
 deny   192.168.100.15
 deny   192.168.100.1
 deny   192.168.100.99
 permit any
 remark Blocheaza accesul serverelor S1, S15 si S99
```

2.4 Translatarea de adrese

2.4.1 Adrese publice și private

Toate adresele IP utilizate în Internet trebuie înregistrate la o autoritate regională. Pentru Europa, aceasta este RIPE; pentru Statele Unite ale Americii, autoritatea este ARIN. Din cele trei clase ale protocolului IP utilizabile pentru adresarea unicast (A, B și C) unele intervale de adrese sunt rezervate pentru moduri speciale de adresare, cum ar fi ruta *default* 0.0.0.0/0 sau adresele loopback din spațiul 127.0.0.0/8 (și altele) iar acestea nu pot fi cumpărate sau re zervate de un anumit client.

Pe lângă adresele rezervate din considerente tehnice, RFC1918 stabilește alte trei intervale de adrese, câte unul din fiecare clasă, al adreselor **private**, ce nu trebuie rutate în Internet. Acestea sunt:

- Pentru clasa A: 10.0.0.0/8 cu intervalul 10.0.0.0 – 10.255.255.255
- Pentru clasa B: 172.16.0.0/12 cu intervalul 172.16.0.0 – 172.31.255.255
- Pentru clasa C: 192.168.0.0/16 cu intervalul 192.168.0.0 – 192.168.255.255

Într-un mediu izolat, al unui laborator, aceste adrese pot fi folosite pentru orice scopuri împreună cu toate celelalte adrese utilizabile ale claselor A,B sau C. Pentru rutarea în Internet, însă, RFC1918 specifică faptul că aceste adrese nu sunt acceptate în antetele pachetelor. Cu alte cuvinte, în Internet nu trebuie să fie permisă adresarea pe baza adreselor private.

Din moment ce aceste adrese sunt rezervate și nu sunt permise în Internet, ele pot fi folosite independent în orice rețea locală; spațiile pot fi împărțite în subrețele după necesități iar utilizarea lor nu necesită niciun fel de aprobare din partea unei autorități. Orice număr de rețele private din lume poate folosi aceste adrese.

Observație: trebuie avut în vedere faptul că aceste adrese nu prezintă niciun fel de proprietăți speciale ce le împiedică a fi utilizate în Internet. RFC1918 specifică doar faptul că **politica** ISP-urilor nu trebuie să permită rutarea lor.

Avantajul major al acestor adrese este gradul de flexibilitate pe care îl permit. Fiecare companie poate să își creeze propria sa schemă de adresare în interiorul său, fără a fi nevoită să obțină aprobări și să cumpere efectiv aceste adrese de la o autoritate regională. De asemenea, aceste adrese permit creșterea neîngrădită a rețelelor chiar și în contextul actual, în care adresele publice disponibile IPv4 devin insuficiente.

2.4.2 Prezentarea conceptelor

Marele dezavantaj al acestor adrese stă în însăși natura lor privată. Din moment ce ele nu pot fi rutate în Internet, rețelele cu adresare privată vor avea nevoie de un mecanism special pentru a le putea oferi o modalitate de a comunica și în exterior. Acest mecanism trebuie sa „traducă" adresele private în adrese publice și să folosească aceste adrese publice pentru încapsularea pachetelor IP provenite din rețele locale, ce vor trebui rutate în Internet. De asemenea, se ridică și problema

traducerii inverse a pachetelor ce vin din Internet şi trebuie să ajungă la anumite destinaţii din reţelele locale.

NAT (Network Address Translation) reprezintă un mecanism care adresează problemele de mai sus. În lipsa NAT, staţiile din reţelele locale nu ar putea avea acces la Internet deoarece adresele lor private nu sunt acceptate ca sursă în mesajele trimise şi nici ca destinaţie în mesajele aşteptate ca răspuns[1].

NAT trebuie să ruleze pe ruterul (sau serverul) ce reprezintă *gateway*-ul spre Internet al unei reţele locale. Ruterul care rulează NAT trebuie să fie configurat cu una sau mai multe adrese publice pe interfaţa conectată la Internet. În momentul în care o staţie din reţeaua locală trimite un mesaj spre Internet, ruterul îi înlocuieşte adresa sursă privată din antetul IP cu o adresă IP publică, de regulă, cea configurată pe interfaţa sa şi reţine această asociere. Pentru staţiile din Internet, comunicaţia va apărea ca şi cum ar avea loc doar între ruterul *gateway* al reţelei şi destinaţia din Internet. Sarcina ruterului care rulează NAT este aceea de a ruta modifica antetul de nivel 3 atât pentru pachetele trimise din LAN cât şi pentru pachetele primite din Internet pe adresa sa publică (rescriindu-le cu adresele private ale staţiilor pentru a putea fi rutate mai departe spre staţiile corespunzătoare).

Fenomenul de rescriere a antetelor descris mai sus are loc cât timp staţia respectivă efectuează trafic spre Internet prin conexiuni iniţiate local. Pachetele venit din Internet vor putea ajunge la staţia corectă doar dacă fac parte dintr-o conexiune iniţiată din LAN. În caz contrar, ruterul nu cunoaşte asocierea între o adresă privată şi o adresă publică şi nu va trimite pachetele spre nicio staţie, deoarece ele vor părea că îi sunt adresate lui.

Deşi imposibilitatea adresării din Internet poate fi de multe ori un dezavantaj, în realitate ea reprezintă o măsură destul de solidă de securitate, comportamentul fiind asemănător unui *firewall* simplu ce nu permite decât conexiunile iniţiate de staţia locală.

Din păcate, mecanismul descris mai sus creează dificultăţi în momentul în care un număr de staţii mai mare decât numărul de adrese publice disponibile încearcă să trimită pachete în Internet la un moment dat, ceea ce e o situaţie întâlnită aproape în permanenţă. Deseori ruterul are o singură adresă publică, iar problema se datorează faptului că ruterul va putea face o singură asociere la un moment dat pentru acea adresă. Dacă două staţii încearcă să comunice în acelaşi timp, ele vor suprascrie asocierea de adrese din ruter în permanenţă, rezultând în pierderi de pachete suferite de către ambele staţii. În mod intuitiv, rezolvarea acestui neajuns poate consta în cumpărarea atâtor adrese publice IP câte sunt necesare pentru ca toate staţiile să comunice în acelaşi timp. Evident, această metodă este extrem de costisitoare, uneori imposibil de aplicat, nu oferă absolut niciun avantaj şi, în plus, introduce un overhead prin utilizarea redundantă a NAT-ului.

Metoda cea mai răspândită de implementare a NAT-ului se numeşte **PAT (Port Address Translation)**. În practică, metoda este atât de răspândită încât referirea tehnologiei NAT, în ziua de astăzi, se referă aproape exclusiv la PAT.

PAT oferă posibilitatea multiplexării conversaţiilor sosite de la staţii diferite din interiorul reţelei locale peste o singură adresă publică. După cum îi spune şi numele, PAT realizează multiplexarea la nivel de porturi, deci la nivelul transport. Un ruter care rulează PAT va suprascrie atât sursa adresei de nivel 3 cât şi sursa adresei de nivel 4 pentru pachetele care pleacă din reţeaua locală şi va suprascrie adresele destinaţiile de nivel 3 şi 4 pentru pachetele-răspuns venite din Internet. Aşadar,

---

[1] În plus, adresarea privată la nivelul Internetului ar fi imposibilă şi datorită faptului că aceleaşi adrese private sunt refolosite în milioane de reţele locale.

un ruter PAT va menține asocieri între perechi de tipul [IP sursă privat, port sursă] şi [IP sursă public, port sursă].

Un ruter ce rulează PAT va suprascrie adresa sursă privată de nivel 3 cu cea publică pentru pachetele trimise în Internet şi va încerca să păstreze acelaşi port sursă din pachetul inițial. Porturile sursă sunt, în marea majoritate a cazurilor, alese la întâmplare de către aplicații, dar sunt extrem de importante pentru că ele informează stațiile-destinație pe ce port să trimită răspunsurile. Ruterul va încerca să conserve acest port, dacăși portul corespunzător al inte rfeței sale publice este liber. În eventualitatea în care două stații încearcă să comunice cu același port sursă, ruterul va înlocui unul dintre ele cu primul găsit liber şi va respecta noua asociere formată.

Ruterul va menține deschise porturile asociate stațiilor pentru a putea primi traficul de răspuns destinat rețelei locale. Datorită asocierilor reținute în memorie, un ruter va şti să direcționeze traficul primit către stațiile corecte deoarece o pereche [adresă IP, număr port] va identifica în mod unic o anumită conexiune de pe o anumită stație din rețeaua locală.

2.4.2.1 Terminologia adreselor în NAT

Datorita modificării adreselor IP în pachetele supuse procesului de NAT, diferite adrese sunt denumite în funcție de originea lor şi de punctul de referință:

- **Inside local**: adrese private, în interiorul rețelei locale;
- **Inside global**: adresa IP publică ce este atașată traficului unei stații din rețeaua locală când iese în Internet prin ruterul NAT. Poate fi chiar adresa publică de pe interfața ruterului.
- **Outside global**: adresa IP publică asignată unei stații din Internet. Această adresă nu se modifică în procesul de NAT deoarece trebuie conservată până când pachetele ajung la destinație.
- **Outside local**: de cele mai multe ori egală cu outside global, este adresa locală a unei stații din Internet. Nu are relevanță în cazul în care NAT se aplică doar adreselor sursă.

2.4.3 Configurarea NAT

2.4.3.1 NAT static

Realizarea de NAT static presupune o mapare 1 la 1 între o adresă privată şi una publică. Sintaxa comenzii este următoarea:

```
R2(config)#ip nat inside source static 192.168.1.1 89.221.57.81
```

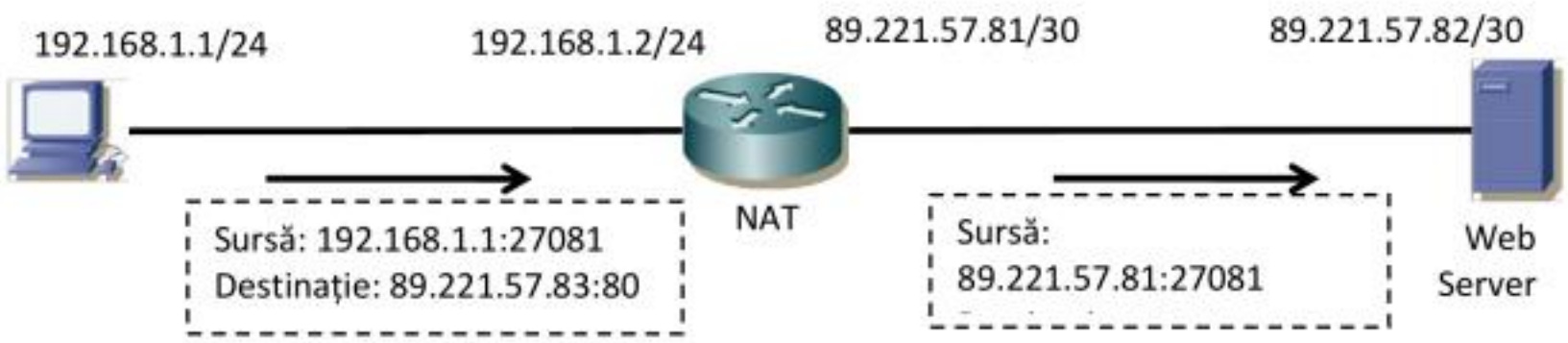

Figura 2-4: NAT static

Comanda introduce o mapare permanentă între adresa locală 192.168.1.1 şi adresa publică 89.221.57.81. Pentru a defini direcția în care se realizează translatarea adreselor, trebuie specificate interfețele inside (spre rețeaua locală) si cele de outside (spre Internet):

```
R2(config)#int fast1/0
R2(config-if)#ip nat inside
R2(config-if)#int fast0/0
R2(config-if)#ip nat outside
```

Comanda de verificare **show ip nat translations** arată asocierea statică definită:

```
R2#sh ip nat trans
Pro Inside global      Inside local       Outside local      Outside
global
--- 89.221.57.81       192.168.1.1        ---                ---
```

2.4.3.2 NAT dinamic

Realizarea de NAT dinamic oferă un surplus de flexibilitate, în sensul că permite definirea adreselor mai multor stații din rețeaua locală, ce vor fi translatate, precum și definirea unei mulțimi de adrese publice în care acestea vor putea fi translatate.

Drept exemplu de configurare, se va considera că toate adresele din rețeaua locală 192.168.100.0/24 vor putea fi translatate la adresele publice din intervalul 200.100.99.226 – 200.100.99.240.

Pentru definirea listei de adrese private ce vor fi supuse procesului de NAT, se creează un ACL ce trebuie să permită doar adresele dorite:

```
R2(config)#access-list 1 permit 192.168.100.0 0.0.0.255
```

Mulțimea adreselor publice se definește prin crearea unui *pool* de adrese ce cuprinde primași ultima adresă:

```
R2(config)#ip nat pool MYPOOL 200.100.99.226 200.100.99.240 netmask
255.255.255.224
```

În final, se configurează procesul NAT să opereze cu ACL-ul si *pool*-ul definite anteriorși se marchează interfețele de *inside* și de *outside*:

```
R2(config)#ip nat inside source list 1 pool MYPOOL
R2(config)#int fast1/0
R2(config-if)#ip nat inside
R2(config-if)#int fast0/0
R2(config-if)#ip nat outside
```

2.4.4 Configurare PAT

Configurarea PAT este similară configurării de NAT dinamic și poate fi realizată în două moduri:

- folosind doar adresa publică de pe interfața conectată la Internet;
- folosind un *pool* de adrese publice.

Pentru ambele cazuri, comanda primește argumentul suplimentar **overload**.

Exemplu folosind adresa IP publică de pe interfață:

```
R2(config)#ip nat inside source list 1 interface FastEthernet 0/0 overload
```

Exemplu folosind un *pool* predefinit de adrese publice:

```
R2(config)#ip nat inside source list 1 pool MYPOOL overload
```

În cazul în care se optează pentru folosirea unui *pool* de adrese publice, adresele suplimentare vor fi utilizate doar pentru cazurile în care apar conflicte cu privire la numerele de port în momentul stabilirii de noi conexiuni.

2.4.5 Port forwarding

Tehnica de *port forwarding* permite accesarea din exterior a unui port de pe o stație dintr-o rețea privată aflată în spatele unui ruter care rulează NAT.

Pentru a putea realiza acest lucru, ruterul NAT trebuie configurat astfel încât să direcționeze toate conexiunile inițiate pe un anumit port al său de pe interfața spre Internet spre un alt port al unei stații din rețeaua internă.

Spre exemplu, o stație cu adresa 192.168.12.13 care rulează un server web într-o rețea locală va putea fi accesată de către ceilalți membri ai rețelei după adresa sa privată și va răspunde la conexiunile inițiate pe portul TCP 80. Deoarece stația se află în rețeaua privată 192.168.0.0/16, ea nu va putea fi accesată din exterior. În consecință, ruterul NAT poate fi configurat astfel încât să redirecteze toate conexiunile externe inițiate pe portul său 80 al interfeței publice spre portul 80 al stației 192.168.12.13, făcând astfel portul din interiorul rețelei accesibil din Internet.

Nu există restricții cu privire la alegerea porturilor ce vor fi *forward*-ate.

Exemplul de mai jos reprezintă modul de configurare al scenariului anterior. Pentru acest exemplu, portul TCP 8080 de pe ruter va fi *forward*-at și va trimite conexiunile spre portul 80 al stației 192.168.12.13 din LAN:

```
R2(config)# ip nat inside source static tcp 192.168.12.13 80 interface
FastEthernet0/0 8080
```

Observație: În general, la configurarea NAT, atunci când există opțiunea de a specifica o adresă publică sau interfața configurată cu acea adresă, se preferă utilizarea variantei bazate pe numele interfeței pentru a nu crea dificultăți în cazul în care adresa IP publică oferită de către ISP se schimbă periodic. Specificând o interfață și nu o anumită adresă IP, ruterul va folosi de fiecare dată adresa IP existentă la momentul respectiv, indiferent de valoarea ei.

3 IPv6

Versiunea a 6-a a protocolului IP a fost introdusă de IETF în anul 1992 ca o propunere de rezolvare a problemei spațiului limitat de adrese al IPv4. În ultimii 17 ani acest protocol a cunoscut o evoluție continuă, pe baza experienței dobândite în perioada de dezvoltare a versiunii a 4-a a protocolului.

Începând cu IPv6 au fost efectuate o serie de modificări în modul de utilizare a unui protocol de nivel 3. În rețele moderne adresarea IP nu se dorește a fi folosită doar pentru laptop -uri și computere, ci se intenționează integrarea unor dispozitive precum: telefoane mobile, televizoare, radio-uri, etc., extinzând conectivitatea la Internet pentru o multitudine de alte dispozitive. O cantitate aproape inepuizabilă de adrese IP este asigurată de IPv6 prin folosirea unei adresări pe 128 de biți.

Deși extinderea substanțială a spațiului de adresare este cel mai cunoscut aspect al protocolului, IPv6 introduce multe alte îmbunătățiri față de predecesorul său. În dezvoltarea sa s-au urmărit identificarea problemelor de funționare, scalabilitate și securitate a versiunii a 4-a și repararea lor folosind soluții simple și eficiente. Printre noile facilități introduse, merită menționate: un antet mult mai simplu aliniat la 64 de biți pentru optimizarea citirii pe noile procesoare x64, configurarea automată a adreselor pe interfață (*autoconfig*), agregarea, securitatea încorporată și capabilitățile de *multihoming*.

Evoluția până în prezent a Internetului şi a adresării bazate pe IPv4 demonstrează faptul că o migrare completă a Internetului la IPv6 pe termen lung este inevitabilă. Migrarea spre un nou protocol este extrem de dificilă şi, mai ales, de durată. De aceea IPv6 propune multiple variante de coexistență cu IPv4. Acestea includ diferite tehnici de tunelare, o formă specială de NAT ce poate translata adrese v4 în adrese v6 și inclusiv rutere ce rulează ambele versiuni ale protocolului și pot comunica prin ambele protocoale.

Trebuie specificat totuși faptul ca tehnologiile IPv4, CIDR și NAT, au reușit să rezolve o serie dintre problemele insuficienței adreselor IPv4, astfel că migrarea la IPv6 va continua ca un proces lent. Schimbarea unui protocol de o asemenea răspândire este un lucru extrem de greu de făcut,

preferându-se întotdeauna varianta care funcționează în prezent și care a reușit să supraviețuiască multor încercări de-a lungul timpului.

Capitolul ce urmează își propune realizarea unei introduceri în lumea protocolului IPv6 cu descrierea caracteristicilor principale, a protocoalelor de rutare cu suport IPv6, iar în încheiere, tehnici de interconectare a rețelelor IPv4 cu rețele IPv6.

3.1 Protocolul IPv6

3.1.1 Antetul IPv6

Pentru a putea înțelege motivația pentru modificările aduse antetului IPv6, se vor analiza mai întâi facilitățile oferite de IPv4.

Ver.	HLEN	ToS	Total Length	
Identification			Flags	Fragment Offset
Time to Live		Protocol	Header Checksum	
Source IP Address				
Destination IO Address				
IP Options (if any)				Padding
Data				
...				

Figura 3-1: Antetul IPv4

În momentul standardizării protocolului IPv4 rețelele erau fundamental diferite față de cele din ziua de astăzi. În primul rând, mediile de transmisie nu aveau aceeași calitate [1] iar din această cauză interferențele din partea mediului erau mult mai mari decât în prezent. O dovadă a acestor probleme din trecut este câmpul *Header Checksum*, folosit pentru verificarea apariției unei erori în antetul IP. În prezent, probabilitatea apariției unei erori de transmisie exact în cei 20 de bytes ai antetului IP este aproape inexistentă însă în continuare fiecare dispozitiv de rețea consumă timp de procesor calculând *checksum*-ul antetului și verificându-l cu cel din pachet. Acesta este un exemplu de câmp al antetului IPv4 ce a devenit complet redundant în prezen, însă există și alte câmpuri ce s-au dovedit a fi ineficiente în comportamentul fluxurilor de date actuale. În această categorie se înscriu și câmpurile folosite pentru fragmentare.

Când un pachet urmează să fie transmis pe o legătură cu un MTU[2] mai mic decât dimensiunea sa, acesta va trebui fragmentat în mai multe pachete. In mod evident, la destinație pachetul inițial va trebui reasamblat. Pentru ca acest lucru să fie posibil, antetul IPv4 un *flag (More Fragments)* de fragmentareși punctele de fragmentare ale datelor folosind câmpurile: *Flags* și *Fragment Offset*. Situațiile în care este nevoie de fragmentare în rețelele actuale sunt foarte puține însă pentru orice transmisie IPv4, câmpul *Fragment Offset* alături de bitul de fragmentare sunt incluse în mod inutil în pachet.

Antetului IPv4 are 12 câmpuri distincte, având în total o dimensiune de 20 *bytes* ce poate fi extinsă prin câmpul *Options* cu dimensiune variabilă.

În contrast, antetul IPv6 păstrează doar 5 câmpuri din cele 12 ale predecesorului său. Acestea sunt aliniate la 64 de biți în vederea optimizării procesării pentru procesoarele pe 64 de biți. Antetul

---

[1] http://en.wikipedia.org/wiki/Category_5_cable

[2] MTU = Maximum Transmission Unit

IPv6 are o structură modular, separând mai clar câmpurile obligatorii de cele opționale sau rar utilizate. Antetul de bază IPv6 are structura de mai jos:

Version	Traffic class	Flow label	
Payload length		Next header	Hop limit
Source address			
Destination address			

Figura 3-2: Antetul de bază IPv6

Lungimea antetului de bază IPv6 este de 40 *bytes* din cauza adreselor sursă și destinație ce ocupă 18 *bytes* fiecare. Câmpurile din figura de mai sus au următoarele roluri:

- **Version**: versiunea protocolului IP; în cazul IPv6 câmpul are mereu valoarea 6.
- **Traffic class**: 8 biți folosiți pentru definirea ToS (*Type of Service*) în implementarea QoS (*Quality of Service*). Funcția acestor biți este comună cu cea din IPv4.
- **Flow label**: este folosit de către sursa unei transmisii pentru a marca identitatea unui flux de date. Alături de adresa sursă și adresa destinație, *label*-ul poate fi folosit în tehnici de QoS sau *Multilayer Switching* pentru a obține un timp mai bun de *switching* al pachetelor.
- **Payload length**: echivalentul lui *Total Length* din IPv4. Este folosit pentru specificarea dimensiunii pachetului.
- **Next header**: echivalentul câmpului Protocol din IPv4. Este folosit pentru specificarea protocolului de nivel superior, încapsulat în pachet.
- **Hop limit**: echivalentul câmpului TTL din IPv4. Este folosit pentru specificarea numărului maxim de hopuri peste care un pachet poate fi rutat. Câmpul este decrementat de către fiecare ruter de pe parcurs, la primirea pachetului.
- **Source address**: adresa IPv6 sursă exprimată pe 128 de biți
- **Destination address**: adresa IPv6 destinație exprimată pe 128 de biți

Pe lângă antetul de bază, IPv6 poate fi extins cu antete opționale utile în diferite scenarii. Printre exemple se numără antetul de fragmentare sau antetul de AH/ESP ce oferă suport integrat IPSec în IPv6.

În general, fragmentarea produsă de IPv4 afectează performanța rețelei de aceea IPv6 introduce o nouă tehnică numită *Path MTU Discovery* (PMD). Folosind această tehnologie, IPv6 poate deduce care este cea mai mică valoarea MTU de la un punct de la celălalt peste o infrastructură de nivel 3. Valoarea determinată va fi folosită pentru toate transmisiile realizate pe acea rută evitându-se astfel fragmentarea pachetului. PMD nu este încă o tehnologie matură, de aceea IPv6 foloseşte în continuare antetul de extensie pentru fragmentare; dacă PMD nu funcționează sau nu poate fi folosit din cauza constrângerilor asupra resurselor de calcul, se atașează acest antet care reține numerotarea și *offset*-ul fragmentelor unui pachet.

IPv6 are, de asemenea, suport nativ pentru IPSec prin protocoalele AH (*Authentication Header*) și ESP (*Encapsulated Security Payload*). AH este folosit numai pentru autentificare iar ESP este folosit în scenariile în care este nevoie de confidențialitatea a datelor. Multe tehnici de autentificare sau securizare folosite anterior în protocoale (ex: autentificarea în protocoale de rutare) s-au mutat acum în IPv6 si pot fi făcute direct peste protocolul de nivel 3.

3.1.2 Reprezentarea adreselor IPv6

Pe orice mediu de transmisie, fie că se folosesc pentru codificare nivele de tensiune (pe cupru), fie că se folosesc semnale de lumină (pe fibră optică), pachetele sunt transmise în unități de biți. Echipamentele de rețea interpretează fluxurile de date la nivel de biți. Odată ce datele au fost

procesate ele sunt reținute în memorie într-un număr de *bytes*, cea mai mică unitate adresabilă de memorie. Pentru un dispozitiv *hardware*, limbajul binar este nativ. Situația este diferită în limbajul uman unde exprimarea unei adrese IPv4 în 32 de biîn notație binară ar fi aproape imposibil de reținut şi dificil de comunicat. Astfel în IPv4 s-a preferat notația zecimală care exprima o adresă IP în doar 4 numere zecimale, cu valori între 0 şi 255.

Deşi sistemul zecimal de exprimare a numerelor era potrivit pentru 32 de biți, la trecerea la IPv6 și adrese de 128 de biți acesta nu a mai fost scalabil.

IETF a hotărât ca modul de reprezentare a adreselor IPv6 să fie făcut în hexazecimal. Reprezentarea hexazecimală are avantajul unei lungimi relativ reduse a adresei IPv6. Un exemplu de astfel de adresă ar fi FE80:0000:00DD:0000:0000:ED01:00F1:1134.

Se poate observa căși în formă hexa o adresă IPv6 este greu de citit și d e reținut. Din acest motiv IETF a adoptat anumite convenții menite să facă o adresă IPv6 mai ușor de citit și exprimat în limbaj uman:

- Literele A-F folosite în sistemul hexazecimal vor fi *case-insensitive* la folosirea într-o adresă IPv6
- Mai multe cifre „0" consecutive folosite la începutul unui grup[1] din adresa IPv6 pot fi omise. Dacă grupul este format numai din cifre „0", acestea pot fi abreviate într-o singură cifră „0".
- Folosirea repetată și continuă a cifrei „0" în mai multe grupuri din adresa IPv6 poate fi abreviată folosind simbolul „::". Această prescurtare poate fi folosită însă o singură dată în exprimarea unei adrese IPv6.

Se va lua ca exemplu adresa de mai sus: FE80:0000:00DD:0000:0000:ED01:00F1:1134.
Aceasta poate fi reprezentată în mai multe forme valide folosind convențiile de mai sus:

- FE80:0:DD:0:0:ED01:F1:1134
- FE8::DD:0:0:ED01:F1:1134
- FE80:0:DD::ED01:F1:1134

Se poate observă că a treia scriere este cea mai scurtă, deci cea mai eficientă. Trebuie avut în vedere faptul că notația „::" poate fi utilizată o singură dată în scrierea unei adrese IPv6. În mod contrar, nu ar mai putea fi determinată forma inițială a adresei[2].

3.1.3 Structura adreselor IPv6

Trei tipuri diferite de adrese IPv6 au fost definite în cadrul RFC-ului 3513: *unicast*, *multicast* și *anycast*. Prima observație este faptul că în IPv6 nu există tipul de transmisie *broadcast*. Evoluția protocoalelor multicast de-a lungul anilor a deschis posibilitatea înlocuirii *broadcast-ului* cu deja existentul *multicast IPv6*.

3.1.3.1 Adresarea anycast

Un nou tip de adresă în IPv6 este *anycast*. Aceasta permite configurarea aceleiași adrese pe mai multe noduri de rețea. Presupunând că pe aceste noduri de rețea rulează un serviciu ce folosește o bază de date comună distribuită, un ruter din Internet poate alege în decizia de rutare calea cea mai scurtă către nodul cel mai apropiat ce oferă respectivul serviciu. Astfel se poate realiza un serviciu

---

[1] Prin definirea unui grup din adresa IPv6 se înțelege o grupare de cifre hexa cuprinse între delimitatorii „:", echivalentul a 2 bytes

[2] Sistemele de operare ce interpretează adrese IPv6 introduc biți de zero în locul structurii "::" până la atingerea totalului de 128 de biți. Cu două astfel de structuri ar fi imposibil de determinat numărul de zerouri ce trebuie introduse în locul fiecăreia.

distribuit în mai multe puncte din Internet iar direcționarea eficientă către serviciu se va putea face de la nivel 3. O aplicație interesantă pentru adresarea *anycast* este în tunelare. Un ruter poate alege în funcție de o adresă *anycast* cea mai rapidă ieșire din tunel.

3.1.3.2 Adresarea unicast

Adresarea *unicast* este elementul fundamental al oricărui tip de transmisie în Internet. O adresă IPv6 are o dimensiune de 128 de biți iar o structură de adresare plată pentru IPv6 ar însemna ca fiecare ruter din Internet să țină toate adresele destinație din lume, o metodă complet nescalabilă. În mod evident, ca și în cazul IPv4, există nevoia unei structuri de adresare ierarhică în IPv6 folosind conceptul de mască de rețea. Din cauza dimensiunii mari ai unei adrese IPv6 singura notație acceptată pentru masca de rețea este folosind simbolul „/" urmat de un număr de la 0 la 128 ce semnifică numărul de biți setați la valoarea „1" de la începutul măștii, ca și în cazul IPv4. Exemplu: 2001:410:0:1::/64.

Adresele unicast pot fi de trei tipuri, în funcție de domeniul lor de „vizibilitate"(*scope*) în rețea:

- *global unicast* – IP-urile din această categorie poartă numele de *global unicast addresses* (*GUAs*)
- *link-local* – IP-urile din această categorie poartă numele de *link-local addresses*
- *unique-local – IP-urile din această categorie poartă numele de unicast local addresses (ULAs)*

Global unicast sunt echivalentul IP-urilor publice în IPv4, fiind rutabile în tot Internetul. Conform RFC-ului 3587, acestor adrese le este rezervat spațiul 200::/3 din totalul de adrese IPv6. Având la dispoziție un număr atât de mare de adrese, unul dintre principalele scopuri ale modului de adresare global a fost realizarea unei scheme de agregare. Cei 128 de biți ai adresei IPv6 au fost împărțiți după cum se observă în figura de mai jos:

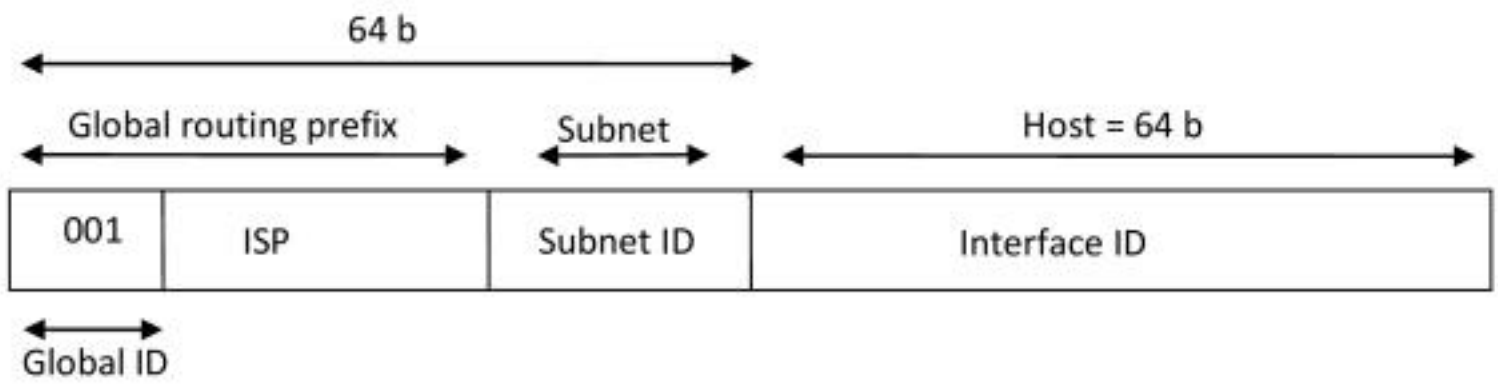

Figura 3-3: Structura adresei IPv6

Primii 3 biți vor avea întotdeauna valoarea „001" pentru a indica o adresă rutabilă conform standardului din RFC. Următorii n biți unde n<45 vor fi identificatorul ISP-ului ce oferă respectivul IP. În funcție de modul în care IANA deleagă adresele ISP-urilor, din primii 48 de biți se poate deduce continentul, țara și mai apoi ISP -ul. Următorii biți, până la 64 (minim 16), vor fi oferiți unei organizații pentru crearea de subrețele. Cu alte cuvinte, porțiunea denumită *Subnet ID* va reprezenta echivalentul părții de subrețea din IPv4.

Ultimii 64 de biți, numiți *Interface ID* în IPv6, vor reprezenta partea de *host* ce va identifica în mod unic o interfață din punct de vedere global. Aceştia pot fi, la rândul lor, împărțiți în subrețele, după necesitățile fiecărei organizații.

În standardul IPv6 nu există o limitare a numărului de adrese ce se pot configura pe o interfață. Față de IPv4, un ruter poate avea în același timp multiple adrese IP pe aceeași interfață incluzând o adrese globale, anycast și link-local.

Adresele *link-local* au un rol foarte important în noua versiune a protocolului IP. Când un dispozitiv IPv6 pornește, el primește automat prin intermediul stivei TCP/IP(v6) o adresă pe care o poate folosi pentru comunicarea pe același *link* cu orice vecin al său. Astfel 2 dispozitive IPv6 nu necesită intervenția administratorului pentru a putea comunica între ele pe același *link*. *Network Discovery Protocol* este un bun exemplu în acest sens. NDP este un protocol ce rezidă în Control Plane[1] în IPv6. El este folosit pentru multe dintre funcționalitățile IPv6 pe un segment de rețea: *autoconfig*, *multicast address resolution* (echivalentul lui ARP în IPv4) etc. Toate mesajele pe care NDP le folosește pentru a ușura configurarea inițială a rețelei folosesc adrese IPv6 *link-local*.

Adresele *link-local* sunt recunoscute ușor deoarece folosesc prefixul FE80::/10. Următorii 54 de biți până la *Interface ID* sunt întotdeauna 0 iar *Interface ID*-ul este generat din adresa MAC folosind metoda EUI-64. Această metodă va fi descrisă în următorul subcapitol: *Network Discovery Protocol*. Un exemplu de adresă link local este: FE80::204:6DFF:FE7F:7C1A/10.

Adresele ULA reprezintă echivalentul adreselor IPv4 private în IPv6 și la prima vedere standardul IPv6 nu ar părea să aibă nevoie de acestea din cauza spațiului imens pe care IPv6 îl pune la dispoziție, adresele private au alte avantaje politice. Încă din IPv4 ele au oferit unei companii un mod de delimitare a rețelei, de creare a unei „zone de confort".

3.1.3.3 Adresarea multicast

Tipul de transmisie multicast joacă un rol extrem de important în IPv6, fiind folosit pentru toate tipurile de transmisii ce utilizau înainte *broadcast*. O adresă multicast poate fi recunoscută ușor prin prefixul FF00::/8. Cel mai bun exemplu pentru înlocuirea *broadcast* cu *multicast* este procesul de *address resolution* realizat în IPv4 prin ARP. Mai multe detalii privind despre funcționarea acestuia vor fi oferite în capitolul ce urmează.

3.1.4 Network Discovery Protocol

NDP a fost pentru prima dată publicat acum 10 ani pentru a introduce în IPv6 capacitatea de autoconfigurare. Cu alte cuvinte, s-a dorit ca la instalarea unei stații noi într-o rețea IPv6, aceasta să se poată configura singură, fără intervenții ale administratorului și să poată avea acces direct la rețea. IPv4 poate realiza aceleași funcții, însă prin utilizarea unui protocol extern: DHCP. În comparație IPv6 oferă funcționalitate *out-of-the-box* fără a avea nevoie de protocoale externe. Este adevărat însă că IPv6 *autoconfig* nu oferă toate avantajele DHCP[2], însă oferă conectivitatea de bază necesară.

Deoarece în IPv6 nu se mai pot folosi transmisii *broadcast*, protocolul ARP ce îndeplinea rolul de realizare a perechilor IP-MAC pentru fiecare stație (*e.g address resolution*) nu se mai poate folosit. Ca și înlocuitor, IPv6 definește NDP care, prin intermediul unor mesaje de tip *Neighbor Discovery* si *Neighbor Solicitation,* realizează cu succes *address resolution* folosind adrese *multicast link-local*.

---

[1] Control Plane – acea componentă a IPv6 care se ocupă cu managementul pachetelor definite de protocol pentru negocierea anumitor parametri. Data Plane se ocupă cu rutarea și comutarea traficului IPv6.

[2] Există o varianta DHCP și pentru IPv6 numită DHCPv6 care preia toate funcționalitățile DHCP și le aduce în lumea IPv6

3.1.4.1 Autoconfig

O interfață poate fi configurată pentru acces la rețeaua locală și globală, în IPv6, în mai multe moduri:

- Automată *stateless*: prin intermediul *IPv6 autoconfig*
- *Manual: prin intermediul intervenției administratorului. În acest tip de configurare administratorul poate aleagă să configureze întreaga adresă IPv6 (rețea+subrețea+interface ID) sau doar partea de rețea și subrețea, lăsând ultimii 64 de biți să fie generați prin metoda EUI-64.*
- *Automată stateful: configurarea prin intermediul DHCPv6*

Pentru IPv6, IETF mai propune o tehnologie *stateless* de configurare a informațiilor pe interfață, numită *DHCP-lite*. Acesta este un server DHCP ce poate oferi doar informații de tipul: adresă server DNS, nume domeniu, adresă server SIP etc. Metoda aceasta nu poate oferi interfeței o adresă IP sau o mască de rețea[1].

Configurarea automată *stateless* este cea care va face subiectul paragrafelor ce urmează. Folosind mesaje de tip *Router Advertisement* si *Router Solicitation* o stație poate comunica cu ruterul *gateway* pentru a obține de la acesta adresa și masca de rețea folosite pe respectivul segment. De asemenea poate instala ruterul respectiv ca adresă de *gateway*. Explicația de nivel înalt a acestui proces este destul de simplă însă în spate se realizează schimbul de mesaje complexe bazate pe adrese *link-local* funcționale și verificate pe baza DAD (*Duplicate Address Detection*).

În acest punct al configurării automate *stateless* stația a obținut de la ruter o adresă de rețea. Rămâne de configurat doar *Interface ID*-ul pentru a putea realiza o comunicație din punct de vedere al nivelului 3.

Generarea *Interface ID*-ului este făcută de către stație pe baza adresei MAC, folosind o metodă standard numită EUI-64. O adresă MAC are 48 de biți. Un *Interface ID* are 64 de biți. Trebuie deci adăugați cumva 16 biți în adresa MAC pentru a obține un *Interface ID* valid. Metoda EUI-64 specifică realizarea acestei operații prin introducerea secvenței „FFFE" la mijlocul adresei MAC.

O adresă MAC 0090AACE0198 va deveni astfel 0090AA**FFFE**CE0198. Administratorul de rețea poate alege implementarea configurării semi-automate prin care el doar specifică primii 64 de biți ai adresei IPv6, urmând ca porțiunea de *host* să fie completată prin EUI-64.

3.1.4.2 Address resolution

IPv6 înglobează protocolul ARP în propria implementare. Prin intermediul NDP și a mesajelor de *Neighbor Discovery* și *Neighbor Solicitation* IPv6 realizează o tabelă de asociere între adrese MACși adrese IP.

În locul adresei de *broadcast* FF:FF:FF:FF:FF:FF, IPv6 folosește o adresă de multicast specială numită *solicited-node multicast*. Acest tip de adresare este derivat din adresa de nivel 3 a destinației și presupune ca orice destinație ce folosește IPv6 să asculte la nivel 2 pentru adresa sa *solicited-node multicast*.

Prefixul unei adrese *solicited-node multicast* este întotdeauna FF02::1:FF00:0000/104 iar ultimii 24 de biți sunt luați din ultimii biți ai adresei IPv6 configurate pe respectiva interfață.

Pentru adresa IPv6 2001:100:ABC:1:0:0:AABB:CCDD/64 adresa *solicited-node multicast* va fi FF02::1:FFBB:CCDD.

---

[1] Pentru mai multe informații despre *DHCP-lite* se recomandă consultarea RFC-ului 3736

3.2 Implementarea IPv6

Unul din avantajele IPv6 este chiar timpul său lung de dezvoltare. Pentru că protocolul IPv4 încă reușește să ofere suport Internetului, Ipv6 are îndeajuns de mult timp pentru a se forma și a fi testat în toate aspectele sale: securitate, mobilitate, tunelare, rutare etc.

În privința capitolului de rutare, în prezent, toate protocoalele de rutare includ suport pentru IPv6.

3.2.1 Rutarea IPv6

Datorită simplității sale, RIP a fost primul protocol care a obținut suport pentru rutare IPv6. Noul protocol de rutare a fost numit *RIPng (next generation*).

RIPng a moștenit simplitatea predecesorului său. Din acest motiv arhitecții RIPng nu au dorit să adauge funcționalități, astfel că RIPng nu aduce îmbunătățiri spectaculoase față de varianta sa IPv4.

Principalele avantaje pe care RIPng le aduce sunt:

- Doi vecini RIPng nu trebuie să fie în același *subnet* pentru a putea comunica rute. Deși condiția ca doua rutere vecine să fie în același *subnet* este valabilă în RIPng după cum era și în RIP, această condiție e deja îndeplinită prin utilizarea adreselor *link-local*;
- Pentru că IPv6 nu folosește *broadcast*-uri, RIPng rulează direct peste multicast, adresa FF02::9;
- În IPv4 securizarea *update*-urilor RIP se face folosind mecanisme de autentificare ale RIP. În IPv6 există deja suport inerent pentru *IPSec*, deci autentificarea vine din protocolul de nivel 3;
- În cadrul RIP nu se puteau rula instanțe diferite ale protocolului pe același *link*. Cisco IOS oferă posibilitatea rulării de 4 instanțe RIP pe același *link*.

Un exemplu de configurare pentru multiple instanțe RIPng va fi prezent în capitolul 4.3.

3.2.2 Interconectarea rețelelor IPv4 cu rețele IPv6

Trecerea de la IPv4 la IPv6 este un proces de durată. Din păcate nu este posibil ca toate companiile și *ISP*-urile să migreze simultan toate rețelele pe IPv6. Trebuie să existe metode prin care o organizație să își poată trece treptat rețeaua pe IPv6 deoarece, până la o migrare completă, se va trece printr-o perioadă în care cele două tipuri de rețele vor trebui să coexiste.

De la primele implementări IPv6 s-a ținut cont de această problemă la care s-au găsit multiple soluții:

- rutere *dual-stack* – acestea pot rula în același timp și IPv4 și IPv6;
- tehnici de tunelare – dacă între două rețele IPv6 există rutere care rulează numai IPv4 va trebui făcut un tunel cu antet IPv4 în care să fie ascuns pachetul IPv6;
- NAT-PT (NAT – *Protocol Translation*) – implementează o metodă de comunicare între o stație ce rulează numai IPv6 și o stație ce rulează numai IPv4.

3.2.2.1 Tunelarea IPv6 peste o rețea IPv4

Soluție care este de departe cea mai folosită în rețelele actuale și asupra căreia se va concentra capitolul de față este tunelarea. Această tehnică presupune încapsularea pachetelor IPv6 cu antete IPv4. Adresele din antetul Ipv4 sunt de fapt adresele sursăși destinație ale ruterelor d e *border* care sunt capetele tunelului. Se va studia un exemplu de tunelare generică pe topologia de mai jos:

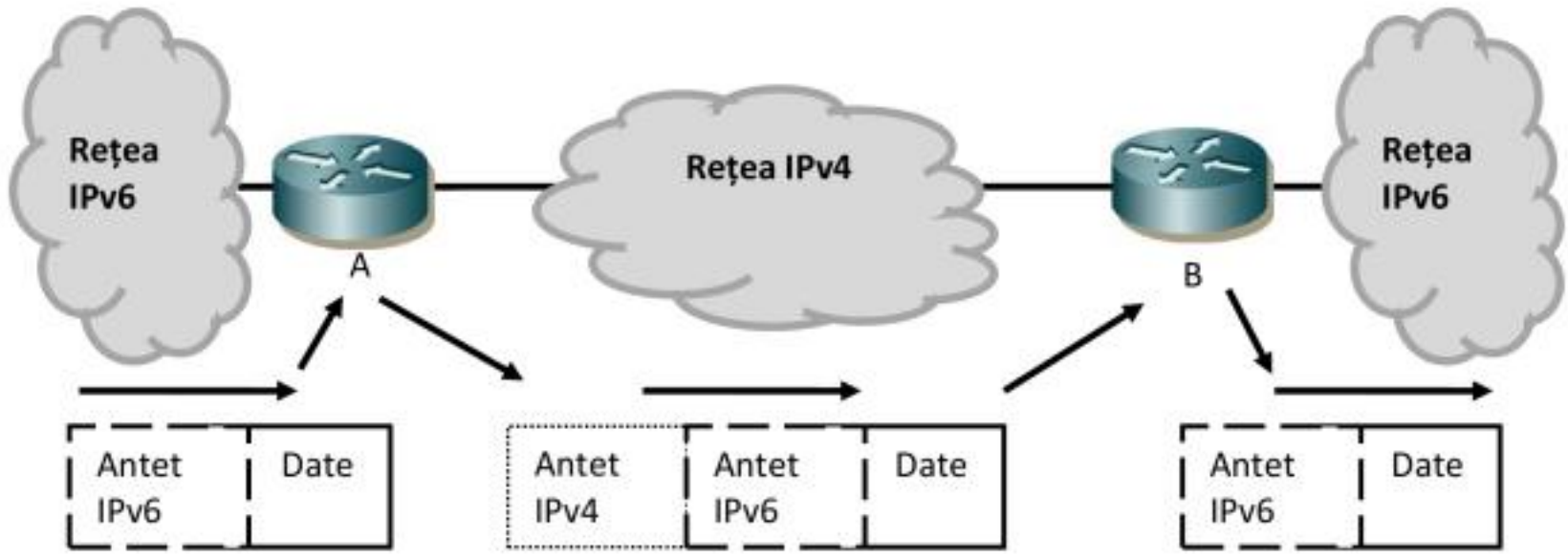

Figura 3-4: Tunelare IPv6 peste o rețea IPv4

Când pachetul cu antet IPv6 ajunge la ruterul A, acesta verifică destinația folosind tabela de rutare. Dacă destinația este cunoscută prin tunel atunci ruterul încapsulează pachetul cu un antet IPv4 astfel:

- Sursă: adresa sa IPv4 dinspre rețeaua IPv4
- Destinație: adresa IPv4 a ruterului B dinspre rețeaua IPv4

După încapsulare pachetul este rutat în Internetul IPv4 până când acesta ajunge la ruterul B. Pentru că pachetul a venit prin tunel, ruterul B va elimina antetul IPv4 și îl va ruta în funcție de tabela de rutare IPv6.

În mare, există două criterii după care se clasifică tunelele în IPv6:

- Dacă tunelul este manual sau automat
- Dacă tunelul este *point-to-point* sau *point-to-multipoint*

Deși există multe tipuri de tunele ce pot fi configurate pentru diferite nevoi de interconectare IPv6 – IPv4, toate sunt construite urmărind câțiva pași de bază:

1. **Asigurarea conectivității IPv4 între capetele tunelului** – dacă nu există conectivitate între adresele cu care pachetul IPv6 este încapsulat, tunelul va fi terminat;
2. **Se creează o interfață de tip Tunnel** – în Cisco IOS introducerea traficului în tunel se face prin crearea unei interfețe virtuale numite *Tunnel (indexată de la 0)*. Pentru a trimite traficul prin tunel și a realiza încapsularea, în tabela de rutare IPv6 trebuie să existe o rută care, pentru o anumită destinație, are interfața tunelului ca interfață de ieșire;
3. **Se configurează sursa tunelului** – adresa IP sursă care va fi folosită pentru încapsularea IPv4. Această adresă trebuie setată în celălalt capăt al tunelului ca adresă IP destinație pentru tunel;
4. **Se configurează o destinație a tunelului (opțional în cazul tunelelor automate)** – adresa IP destinație a tunelului; reprezintă adresa IP sursă a capătului celălalt de tunel;
5. **Configurarea unei adrese IPv6 pentru interfață de tip Tunel** – interfața de tip tunel funcționează ca o prelungire a rețelei IPv6 și de aceea trebuie să aibă o adresă IPv6 dintr-un *subnet* separat;
6. **Configurarea modului de tunelare** – în acest ultim pas se configurează tipul de tunel: MCT, GRE, 6to4, ISATAP etc.

Tunelele *point-to-point* precum MCT (*Manually Configured Tunnel*) sau GRE (*Generic Router Encapsulation*) trebuie configurate pentru fiecare pereche de sursă – destinație. Cu alte cuvinte

pentru fiecare tunel trebuie creată o nouă interfață de tunel în ambele capete. În anumite situații astfel de tunele nu scalează, fiind nevoie de mult efort administrativ pentru configurarea lor.

Soluția în aceste cazuri este utilizarea tunelelor automate. Aceste tipuri de tunele nu au nevoie de specificarea destinației deoarece au încorporat în protocol o metodă prin care pot să o determine automat. Efortul de configurare se reduce astfel la crearea unei singure interfețe de tunel în care trebuie specificată doar sursa tunelului indiferent de numărul de tunele necesar. Astfel de tunele poartă numele de *point-to-multipoint*. În cele ce urmează se va studia modul de funcționare al unui tunel automat *point-to-multipoint*: **Tunelul 6to4**.

3.2.2.2 Tunelarea 6to4

Dacă se dorește un tunel automat și scalabil prima observație este că trebuie să existe o metodă de „descoperire" a adresei IP destinație cu care trebuie încapsulat pachetul.

Pentru a avea această funcționalitate, tunelele 6to4 definesc un prefix special ce trebuie folosit în cadrul rețelelor IPv6 ce doresc să comunice: 2002::/16. Orice „insulă IPv6" ce dorește să comunice cu altă insulă trebuie să aibă adrese IPv6 create astfel:

2002	32 de biți ai adresei IPv4 sursă a tunelului	Subnet	Interface ID

Figura 3-5: Formatul unei adrese 6to4

După cum se observă din figura de mai sus, o adresă 6to4 trebuie creată folosind cei 32 de biți ai adresei sursă de tunel. Adresa IPv6 de tunel pentru fiecare dintre cele două capete ale tunelului trebuie să folosească acest format de adresare pentru a fi posibilă extragerea adresei IPv4, folosită de tunel la încapsulare. Cei 16 biți de *subnet* pot fi folosiți în interiorul fiecărei insule pentru a aloca adrese în același format cu primii 48 de biți comuni în toate rețelele. În continuare se va analiza un scenariu simplu pentru a evidenția comportamentul unui tunel 6to4.

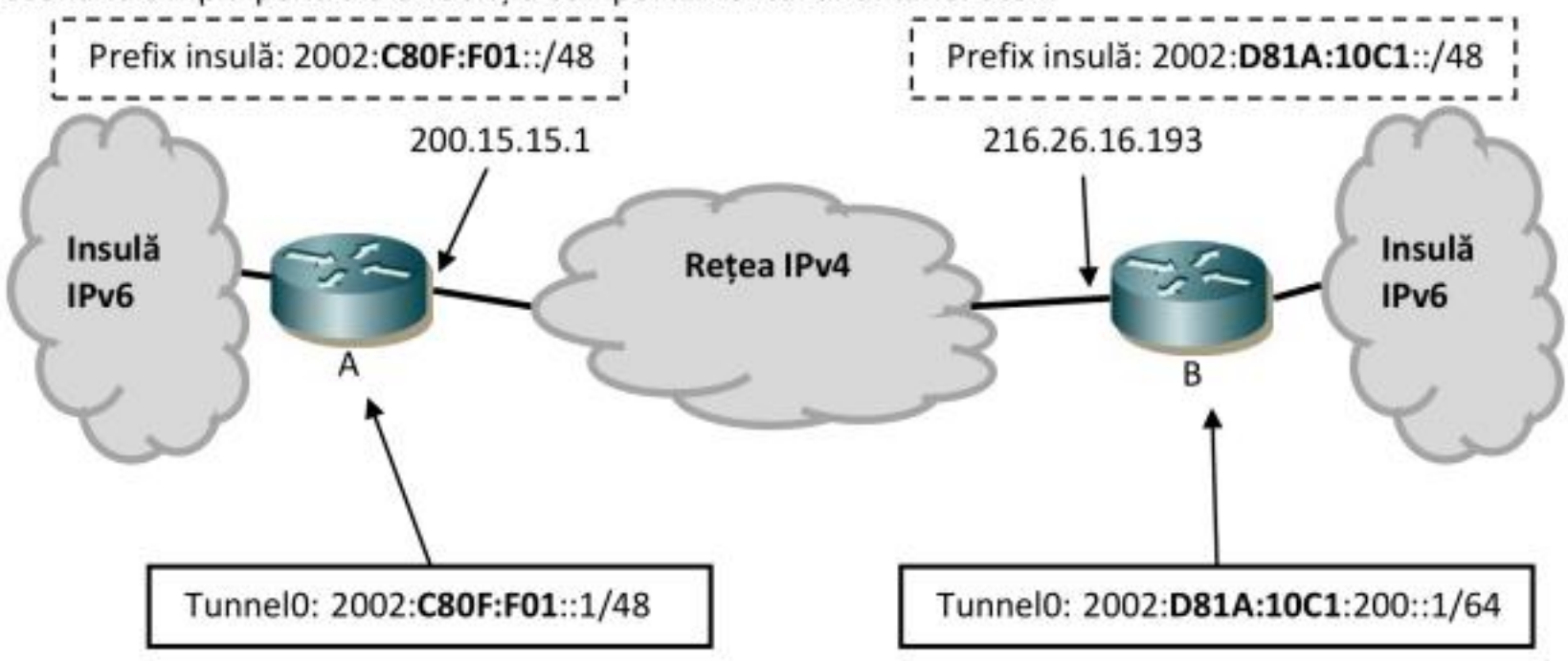

Figura 3-6: Tunelarea 6to4

În afară de configurațiile din figură fiecare ruter mai are o rută spre `2002::/16` care are ca interfață de ieșire Tunnel0. Se va presupune că o stație din insula legată la ruterul A va trimite un ping către o stație din insula B. Pentru că toate subrețele din spatele ruterului A folosesc aceiași 48 de biți (sunt diferențiate prin biții de subnet) pachetul va pleca cu adresă IPv6 sursă:

2002:C80F:F01::X. Când va ajunge la ruterul A acesta îl va trimite prin interfața de tunel. Înainte de a putea însă trimite pachetul în tunel, ruterul A trebuie să realizeze încapsularea IPv4 de tunel. Adresa IP sursă este configurată pe tunel însă adresa IP destinație trebuie dedusă. Ruterul analizează adresa IPv6 destinație a pachetului și extrage cei 32 de biții dintre bitul 16 și bitul 48 care reprezintă adresa IPv4 a celuilalt capăt de tunel. Pachetul este acum încapsulatși poate fi trimis peste rețeaua IPv4.

În mod asemănător se realizează și încapsularea pentru traficul de întoarcere pe ruterul B. Configurațiile pentru tunele MCT, GRE și 6to4 vor putea fi găsite în subcapitolul Studiu de Caz.

3.3 Configurarea IPv6

3.3.1 Configurarea adresării

În mod implicit o interfață de ruter *dual-stack* rulează doar IPv4. Prin aceasta se înțelege că nu va procesa pachetele trimise pe adresele de multicast FF02::1, FF02::2 și adresa de *solicited-node multicast*. De asemenea, nu este posibilă rutarea pachetelor ce folosesc antete IPv6.

Pentru a activa o interfață în protocolul IPv6 se folosește comanda **`ipv6 enable`**. Odată cu activarea se generează și o adresă *link-local* după algoritmul descris în primul capitol.

```
Leshrac(config)#int Ethernet 2/1
Leshrac(config-if)#ipv enable
Leshrac(config-if)#^Z
Leshrac#sh ipv int Ethernet 2/1
Ethernet2/1 is up, line protocol is up
  IPv6 is enabled, link-local address is FE80::CE01:AFF:FE1C:21
  No global unicast address is configured
  Joined group address(es):
    FF02::1
    FF02::2
    FF02::1:FF1C:21
  MTU is 1500 bytes
  ICMP error messages limited to one every 100 milliseconds
  ICMP redirects are enabled
  ND DAD is enabled, number of DAD attempts: 1
  ND reachable time is 30000 milliseconds
  ND advertised reachable time is 0 milliseconds
  ND advertised retransmit interval is 0 milliseconds
  ND router advertisements are sent every 200 seconds
  ND router advertisements live for 1800 seconds
  Hosts use stateless autoconfig for addresses.
```

Același efect ca și cel de mai sus poate fi obținut prin setarea unei adrese IPv6 pe interfață. După cum se poate vedea mai jos, un ruter acceptă mai multe adrese IPv6 pe aceea și interfață:

```
Leshrac(config)#int ethernet 2/1
Leshrac(config-if)#ipv add 2001::1/64
Leshrac(config-if)#ipv add 2001:100::2/64
Leshrac(config-if)#ipv add 2001:200::3/64
Leshrac(config-if)#^Z
Leshrac#sh ipv int br
[...]
Ethernet2/1                 [up/up]
    FE80::CE01:AFF:FE1C:21
    2001::1
    2001:100::2
    2001:200::3
[...]
```

Pentru a activa NDP și rutarea de pachete IPv6, trebuie folosită comanda:

```
Leshrac(config)#ipv6 unicast-routing
```

În primul capitol de *Design* s-au discutat modurile prin care un administrator poate seta adrese IP pe o interfață IPv6. Unul dintre acestea era metoda semi -automată prin care administratorul specifică adresa de rețea și partea de *subnet* însă lasă ruterul să își genereze propriul *Interface ID* unic pe legătură.

```
Leshrac(config-if)#ipv address 2001:400::/64 eui-64
Leshrac(config-if)#do sh ipv int Ethernet2/1
Ethernet2/1 is up, line protocol is up
[…]
    2001:400::CE01:AFF:FE1C:21, subnet is 2001:400::/64 [EUI]
[…]
```

Fiind date două rutere legate printr-o conexiune *point-to-point,* folosind *Stateless auto-configuration* se poate configura automat interfața unui ruter folosind informațiile pe care vecinul său le poate furniza despre adresa și masca de rețea a legăturii respective.

```
Sange(config)#int e2/1
Sange(config-if)#ipv6 address autoconfig
Sange(config-if)#do sh ipv int e2/1
Ethernet2/1 is up, line protocol is up
  IPv6 is enabled, link-local address is FE80::CE0D:AFF:FE1C:21
  Global unicast address(es):
    2001::CE0D:AFF:FE1C:21, subnet is 2001::/64 [PRE/TEN]
```

Pentru a configura o adresă *anycast* administratorul trebuie să specifice explicit acest fapt ruterului deoarece adresele *anycast* sunt extrase din spațiul adreselor unicast.

```
Sange(config-if)#ipv address 2003::1/64 anycast
```

3.3.2 Configurări de rutare

Rutarea statică funcționează în IPv6 în același mod ca și în IPv4 cu excepția definirii unei rute pe un mediu multiacces dată prin interfața de ieșire. În IPv4, la definirea unei astfel de rute, posibilitatea de a trimite pachete pe acea interfață depindea de capacitatea ruterului vecin de a face Proxy ARP. Cum în IPv6 nu mai există ARP, toate rutele statice date pe un mediu multiacces (Ethernet) trebuie date cu *next-hop* în locul interfeței de ieșire.

```
Leshrac(config)# ipv route 2002::/16 2003:100::1
```

În toate protocoalele de rutare IPv6 configurațiile la nivelul modului **(config-router)#** s-au mutat în modul **(config-if)#**. Principalul element adus protocolului RIPng ce se reflectă în configurații este posibilitatea de a rula până la 4 instanțe diferite de RIPng pe aceeași legătură. Instanța se definește prin intrarea în modul **(config-router)#** iar activarea se face la nivel de interfață prin asocierea cu un parametru ce definește instanța.

```
RouterR3#configure terminal
RouterR3(config)#ipv6 router rip RIPNGR3
RouterR3(config-rtr)#int fastethernet0/1
RouterR3(config-if)#ipv6 address 2001:410:ffff:1::1/64
RouterR3(config-if)#ipv6 rip RIPNGR3 enable
RouterR3(config-if)#ipv6 rip default-information originate
RouterR3(config-if)#int fastethernet0/0
RouterR3(config-if)#ipv6 address 2001:410:ffff:2::1/64
RouterR3(config-if)#ipv6 rip RIPNGR3 enable
RouterR3(config-if)#ipv6 rip default-information originate
RouterR3(config-if)#exit
```

În exemplul de mai sus instanța RIPNGR3 este activată pe interfețele fa0/1 și fa0/0. De asemenea ruterul vecin va trimite rute *default* pe aceste interfețe ca rezultat al comenzii **ipv6 rip default-information originate**.

3.3.3 Configurări de interconectare IPv4-IPv6

Se va prezenta mai întâi configurarea unui tunel MCT (*Manually Configured Tunnel*). Acesta este cel mai simplu tunel *point-to-point* ce poate realiza interconectarea de rețele IPv6 peste un Internet IPv4. Deoarece acest tip de tunel nu este o soluție scalabilă, el este folosit pentru a conecta un număr limitat de rețele IPv6.

Urmând pașii descriși în capitolul 4.2.2.1 se va prezenta configurarea unui astfel de tunel în cele două capete ale sale.

```
Leshrac(config)# interface Tunnel0
Leshrac(config-if)# no ip address
Leshrac(config-if)# ipv6 address 2001::1/64
Leshrac(config-if)# tunnel source Ethernet2/1
Leshrac(config-if)# tunnel destination 172.1.231.1
Leshrac(config-if)# tunnel mode ipv6ip

Sange(config)# interface Tunnel0
Sange(config-if)# no ip address
Sange(config-if)# ipv6 address 2001:200::1/64
Sange(config-if)# tunnel source Ethernet2/1
Sange(config-if)# tunnel destination 172.1.231.2
Sange(config-if)# tunnel mode ipv6ip
```

Din output-ul de mai jos se poate observa că adresa tunel destinație configurată la un capăt este adresa de pe *tunnel source* al celuilalt capăt.

```
Leshrac#sh ip int br Eth 2/1
Interface                  IP-Address      OK? Method Status
Ethernet2/1                172.1.231.2     YES NVRAM  up

Sange#sh ip int br E2/1
Interface                  IP-Address      OK? Method Status
Ethernet2/1                172.1.231.1     YES NVRAM  up
```

Până în acest punct ambele interfețele de tunel au fost configurate corect și sunt în starea „up". Totuși, când se încearcă trimiterea unui ping dintr-o interfață de tunel în cealaltă, aceasta nu funcționează. Problema este foarte ușor de intuit: nu există rută în tabela de rutare IPv6. După adăugarea unei rute pe fiecare dintre rutere pentru rețelele IPv6 *remote*, ping-ul va funcționa.

```
Sange(config)#ipv route 2001::/64 Tunnel0
Leshrac(config)#ipv route  2001:200::/64 Tunnel0
```

```
Sange#ping ipv 2001::1 source Tunnel 0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2001::1, timeout is 2 seconds:
Packet sent with a source address of 2001:200::1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/33/96 ms
```

Soluția de tunel MCT sau GRE este cea mai bună când se pune problema unei rezolvări rapide. Pentru soluții permanente și scalabile sunt preferate tunelele automate.

În continuare se vor prezenta configurațiile realizate pentru instalarea unui tunel 6to4 între două insule IPv6.

Mai întâi se vor configura adresele IPv4 pe interfețele legate la rețeaua IPv4.

```
Sven(config)#interface FastEthernet3/0
Sven(config-if)#ip address 172.14.31.2 255.255.255.252

Leshrac(config)#interface Ethernet2/1
```

```
Leshrac(config-if)#ip address 172.1.231.2 255.255.255.252
```

Pentru ca metoda automată de descoperire a adreselor IP destinație din încapsulare să funcționeze, va trebui să obținem echivalentul în hexazecimal al fiecărei adrese IP:

- 172.14.31.2 -> AC0E:1F02 -> Adresa IPv6 pentru Sven va fi 2002:AC0E:1F02:xxxx::x/64
- 172.1.231.2 -> AC01:EF02 -> Adresa IPv6 pentru Leshrac va fi 2002:AC01:EF02:xxxx::x/64

Configurațiile de tunel sunt următoarele:

```
Leshrac(config)# interface Tunnel0
Leshrac(config-if)# no ip address
Leshrac(config-if)# no ip redirects
Leshrac(config-if)# ipv6 address 2002:AC01:EF02:2::2/64
Leshrac(config-if)# tunnel source Ethernet2/1
Leshrac(config-if)# tunnel mode ipv6ip 6to4

Sven(config)# interface Tunnel0
Sven(config-if)# no ip address
Sven(config-if)# no ip redirects
Sven(config-if)# ipv6 address 2002:AC0E:1F02:4::2/64
Sven(config-if)# tunnel source FastEthernet3/0
Sven(config-if)# tunnel mode ipv6ip 6to4
```

Tot ce trebuie adăugat pentru ca a face posibilă comunicarea între cele 2 rutere este o rută 2002::/16 cu interfață de ieșire Tunnel0 pe fiecare dintre capetele tunelului.

3.4 Studiu de caz

În studiul de caz ce urmează se va realiza proiectarea și instalarea a 3 rețele IPv6 începând de la o infrastructură IPv4. În final va trebui ca toate rețele IPv6 să poată comunica între ele peste IPv4.

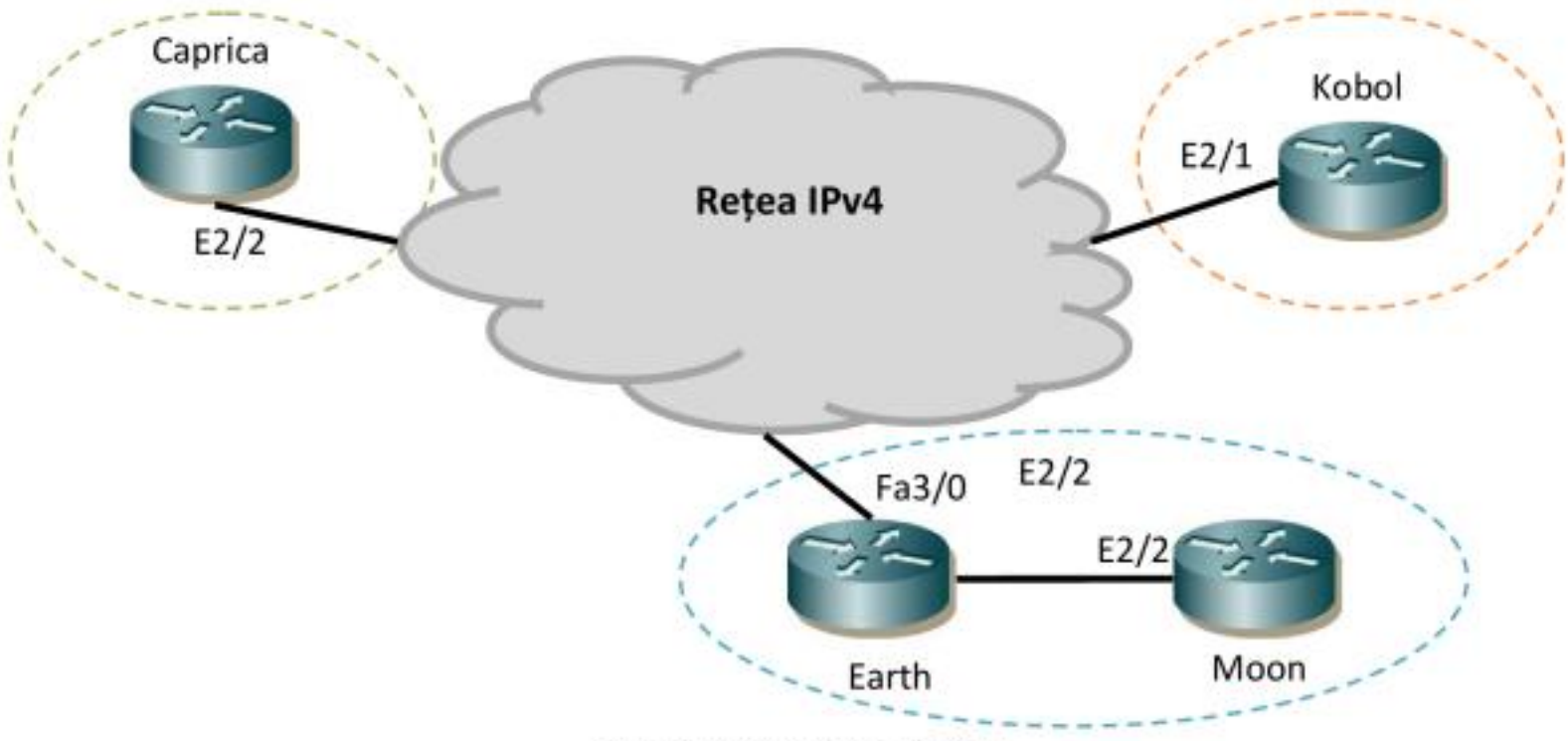

Figura 3-7: Studiu de caz

Primul pas în configurarea rețelei îl constituie insula creată din ruterele Earth și Moon. Rețeaua dintre Earth și Moon trebuie configurată cu adrese IPv6 din spațiul 2001:100::/64. Earth trebuie configurat manual în timp ce Moon trebuie să obțină parametrii de nivel 3 prin *stateless autoconfig*.

```
Earth(config)#int ethernet 2/2
Earth(config-if)#ipv6 address 2001:100::1/64
Moon(config)#int e2/2
Moon(config-if)#ipv address autoconfig

Earth#sh ipv int br
[…]
Ethernet2/2                [up/up]
    FE80::CE06:8FF:FE74:22
    2001:100::1

Moon#sh ipv int br
[…]
Ethernet2/2                [up/up]
    FE80::CE0F:8FF:FE74:22
    2001:100::CE0F:8FF:FE74:22
```

Ruterele Caprica și Kobol vor deține fiecare câte o interfață de loopback prin care o să simuleze o rețea Ethernet:

- Caprica: Loopback1 -> 2001:3::1/64
- Kobol: Loopback1 -> 2001:4::1/64

```
Caprica#sh ipv int br Lo1
Loopback1                  [up/up]
    FE80::CE09:8FF:FE74:0
    2001:3::1
Kobol(config)#int lo1
Kobol(config-if)#ipv address 2001:4::1/64
Kobol(config-if)#exit
Kobol(config)#do sh ipv int br Lo1
Loopback1                  [up/up]
    FE80::CE01:8FF:FE74:0
    2001:4::1
```

Deoarece se dorește implementarea unei soluții scalabile administratorul de rețea a luat decizia utilizării unui tunel automat 6to4. Următorul pas în configurare va necesita traducerea în hexazecimal a adreselor IPv4 de pe ruterele Earth, Kobol, Caprica. Adresele IPv4 corespunzătoare sunt:

```
Earth#sh ip int br fa3/0
Interface                  IP-Address      OK? Method Status
FastEthernet3/0            172.14.31.2     YES NVRAM  up

Kobol# sh ip int br e2/1
Interface                  IP-Address      OK? Method Status
Ethernet2/1                172.1.231.2     YES NVRAM  up

Caprica#sh ip int br e2/2
Interface                  IP-Address      OK? Method Status
Ethernet2/2                172.17.0.2      YES NVRAM  up
```

Pentru fiecare dintre cele 3 rutere traducerile vor fi:

- Earth: AC0E:1F02
- Kobol: AC01:E702
- Caprica: AC11:0002

Odată obținuți biții 16-48 din fiecare adresă IPv6, se pot configura interfețele de tunel cu adresele IP corespunzătoare.

```
Kobol(config)# interface Tunnel0
Kobol(config-if)# ipv6 address 2002:AC01:EF02:2::2/64
Kobol(config-if)# tunnel source Ethernet2/1
Kobol(config-if)# tunnel mode ipv6ip 6to4

Caprica(config)# interface Tunnel1
```

```
Caprica(config-if)# ipv6 address 2002:AC11:0002:2::1/64
Caprica(config-if)# tunnel source Ethernet2/2
Caprica(config-if)# tunnel mode ipv6ip

Earth(config)# interface Tunnel0
Earth(config-if)# ipv6 address 2002:AC0E:1F02:4::2/64
Earth(config-if)# tunnel source FastEthernet3/0
Earth(config-if)# tunnel mode ipv6ip 6to4
```

Interfețele de tunel sunt în acest moment în starea „up" și sunt pregătite pentru rutare. Singura componentă care lipsește sunt rutele statice care să introducă traficul IPv6 în tunel. Pentru a comunica între interfețele de tunel avem nevoie de rute către rețeaua 2002::/16.

```
Kobol(config)# ipv6 route 2002::/16 Tunnel0
Earth(config)# ipv6 route 2002::/16 Tunnel0
Caprica(config)# ipv6 route 2002::/16 Tunnel0
```

După cum se poate vedea în rezultatul de mai jos, comunicația între interfețele de tunel este acum posibilă.

```
Earth#ping 2002:AC11:0002:2::1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2002:AC11:2:2::1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 16/71/144 ms
```

Scopul final este însă posibilitatea de a da ping în interfețele de Loopback ale ruterelor Caprica și Kobol și în rețeaua IPv6 din spatele ruterului Earth.

Toate aceste rețele au aceiași 16 biți deci, teoretic, problema s-ar putea rezolva prin instalarea unei rute 2001::/16 pe fiecare ruter de *border*, având interfața tunelului drept interfață de ieșire. Problema cu această soluție este că ruterele de *border* nu vor ști cu ce IP destinație să încapsuleze pachetului când îl introduc în tunel deoarece ping-ul se dă într-o adresă 2001::/16 și nu într -o adresă 6to4. Pentru a face această schemă să funcționeze trebuie să se ofere fiecărui ruter de *border* în mod explicit un *next-hop* din spațiul 6to4.

Acest lucru se poate face cel mai simplu desemnând unul dintre trei rutere ca *hub* 6to4 și configurând pe celelalte 2 rute statice cu *next-hop*, *hub-ul*. Astfel cele 2 rutere vor deduce IP-ul destinație pentru încapsulare din ruta statică, iar *hub*-ul va realiza rutarea între adrese 6to4.

Se va considera ruterul Earth ca fiind *hub*-ul topologiei și se vor defini rute statice pe Caprica și Kobol. De asemenea, se va configura o rută default pe Moon pentru ca acesta să poată avea acces la celelalte insule.

```
Caprica(config)# ipv6 route 2001::/16 2002:AC0E:1F02:4::2
Kobol(config)# ipv6 route 2001::/16 2002:AC0E:1F02:4::2
Moon(config)# ipv6 route ::/0 2001:100::1
```

Rezultatul final este faptul că un ping de pe Moon pe interfața de *loopback* a lui Kobol va demonstra conectivitatea.

```
Moon#ping 2001:4::1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 2001:4::1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/46/156 ms
```

4 EIGRP

4.1 Prezentarea protocolului

Enhanced Interior Routing Protocol (EIGRP) este un protocol de rutare proprietar Cisco. EIGRP a fost descris ca fiind un protocol de rutare hibrid, deoarece combină caracteristicile protocoalelor bazate pe vectori de distanță cu algoritmii protocoalelor bazate pe starea conexiunilor. Totuşi, o descriere mai corectă ar fi cea de protocol avansat bazat pe vectori de distanță.

Față de protocoalele bazate pe vectori de distanță RIP v1 şi IGRP, EIGRP oferă: convergență rapidă, consum redus de lățime de bandă, suport pentru protocoale rutate multiple şi suport pentru CIDR şi VLSM. Aceste caracteristici sunt obținute prin folosirea unor mecanisme specifice protocoalelor bazate pe starea legaturii, cum ar fi actualizare parțială şi de descoperire a vecinilor.

O limitare însemnată provine din natura proprierară a protocolului, EIGRP fiind o soluție de rutare pentru rețele ce conțin doar echipamente Cisco. A doua limitare o reprezintă natura neierarhică a protocolului, scalabilitatea protocolului recomandându-l pentru implementări de sub 1000 de rutere.

Cu toate acestea EIGRP oferă, cel mai adesea, timpi de convergență mai reduşi decât OSPF, mai ales în cazul unui grad ridicat de redundanță. Acest lucru este realizat prin identificarea unor rute viabile, nu doar a celor mai bune căi precum în cazul protocoalelor clasice de rutare.

În plus, complexitatea de configurare şi administrare a EIGRP este semnificativ mai scăzută decât cea impusă de către OSPF.

4.1.1 Caracteristici ale EIGRP

Cele mai importante caracteristici ale protocolului de rutare EIGRP sunt: convergența rapidă, suportul pentru VLSM, actualizările parțiale şi suportul pentru protocoale rutate multiple.

Convergența rapidă este dată de faptul ca ruterul stochează tabelele de rutare ale tuturor vecinilor, iar atunci cand o rută dispare, va folosi o rută alternativă. În cazul în care nu are o astfel de rută, îşi va întreba vecinii dacă cunosc o rută alternativă. Întrebarea se propagă până când este descoperită o rută, sau se determină ca o rută alternativă nu există.

Suportul pentru VLSM permite folosirea măştilor de lungime diferită pentru aceeaşi rețea şi suportul pentru subrețele discontinue. EIGRP este un protocol de rutare *classless*, ceea ce înseamnă că atunci când anunță o rută către o rețea, trimite şi masca de rețea.

Spre deosebire de celelalte protocoale bazate pe vectori de distanță, EIGRP nu trimite actualizări periodice, ci actualizări parțiale, declanşate de evenimentele din rețea. Actualizările sunt trimise numai când calea sau metrica pentru o anumită destinație se schimbă, şi conțin doar

informații despre rutele modificate. EIGRP are un comportament diferit față de protocoalele bazate pe starea legăturilor, acestea din urmă trimițând actualizările către toate ruterele din rețea. EIGRP foloseşte pachete unicast şi multicast pentru a propaga actualizarea doar către ruterele care au nevoie să cunoască acea informație. Toate aceste mecanisme duc la un consum mai mic de lățime de bandă.

EIGRP deține o serie de componente specifice, care îl diferențiază de celelalte protocoale de rutare: module indepedente de protocol, *Reliable Transport Protocol (RTP)*, mecanismul de descoperire a vecinilor şi *Diffusing Update Algorithm (DUAL)*.

Modulele independente permit ca EIGRP să suporte mai multe protocoale de nivel rețea, precum IP, AppleTalk şi IPX. Aceste module sunt responsabile cu cerințele specifice nivelului rețea. *RTP* va garanta livrarea ordonată a pachetelor EIGRP tuturor vecinilor, deşi, din motive de eficiență, doar anumite pachete vor avea livrarea garantată. *RTP* foloseşte o combinație între transmisia unicast şi multicast.

4.1.2 Descoperirea vecinilor EIGRP

Există două probleme în legatură cu actualizările sigure folosite de EIGRP. Pe de o parte, ruterul nu ştie câte pachete de tip *Ack* să aştepte, deoarece nu cunoaste numărul de echipamente rutere din rețea. Pe de altă parte, ruterul nu ştie dacă o actualizare pierdută înseamnă că nu există nici o informație nouă, sau că vecinul este deconectat.

Aceste probleme sunt adresate de către protocolul EIGRP folosind noțiunea de vecini. Pachetele de tip *Hello* sunt folosite pentru a construi o tabelă de vecini pe fiecare ruter. Dacă un vecin nu trimite pachete *Hello* o anumită perioadă de timp (*hold time*), atunci vecinul este scos din tabela EIGRP şi se produce o reconvergență a rutării.

Un proces EIGRP începe cu descoperirea vecinilor. Apoi, va trimite actualizări în mod multicast şi va aştepta confirmări unicast. Tabela cu vecinii va fi folosită pentru a verifica care dintre aceştia nu au confirmat. Ruterul va trimite actualizările în mod unicast tuturor vecinilor care nu au confirmat. După 16 actualizări trimise către un vecin fără a primi o confirmare, ruterul îl va şterge din tabela de vecini. Atunci când vecinul este reconectat şi trimite din nou pachete Hello, va fi reintegrat in procesul de selectare a căilor optime.

4.1.3 Calculul metricii EIGRP

EIGRP foloseşte o metrică compusă asemănătoare celei folosite de IGRP. Această metrică are următoarele componente: *lățimea de bandă (bandwidth), latența (delay), siguranța (reliability)* şi *încărcarea (load)*. Singura diferență între cele două protocoale este că metrica EIGRP este egală cu metrica IGRP înmulțită cu 256, deoarece se folosesc 32 de biți în loc de 24.

Componentele privind lățimea de bandă şi latența sunt considerate în mod implicit în calculul metricii. *Bandwidth* se referă la cea mai mică lățime de bandă de pe calea către destinație. *Delay* se referă la suma latențelor de pe fiecare legătură din calea către destinație. În plus, pot fi avute în vedere în calculul metricii şi variabile privind siguranța şi încărcarea. *Reliability* este calculată ca fiind cel mai slab nivel de siguranță de pe calea către destinație, fiind estimată prin schimbul de mesaje *keepalive*. Load se referă la cel mai defavorabil nivel de încărcare de pe calea către destinație, estimat prin rata pachetelor şi prin lățimea de bandă configurată.

Formula completă pentru calculul metricii este:

metrica = 256 x [K1 x Bandwidth + (K2 x Bandwidth)/(256 – Load) + K3 x Delay] x [K5/(Reliability + K4)]

Această formulă este folosită pentru valori nenule a factorului K5. În cazul în care K5 este 0 formula de calcul a metricii este:

metrica = 256 x [K1 x Bandwidth + (K2 x Bandwidth)/(256 – Load) + K3 x Delay]

În mod implicit K2=K4=K5=0 şi K1=K3=1. În această situaţie singurii factori ce intră în componenţa metricii sunt lăţimea de bandă şi latenţa interfeţei. Formula de calcul a metricii devine:

metrica = 256 x (Bandwidth + Delay)

Parametrii K pot fi modificaţi, dar este necesară o planificare atentă, deoarece modificarea acestor valori poate cauza probleme de convergenţă.

Bandwidth este calculată după formula Bandwidth = 10^7 / min(bandwith), unde min(bandwidth) înseamnă lăţimea de bandă minimă întâlnită pe calea către destinaţie, exprimată în Kbps.

Delay este calculată după formula Delay = ∑delays / 10, unde ∑delays înseamnă suma latenţelor legăturilor de pe calea către destinaţie, exprimate în microsecunde.

De exemplu, în topologia din figura 4-1,lăţimile de bandă sunt reprezentate în figură iar latenţa pentru fiecare legătură este egală cu 20000.

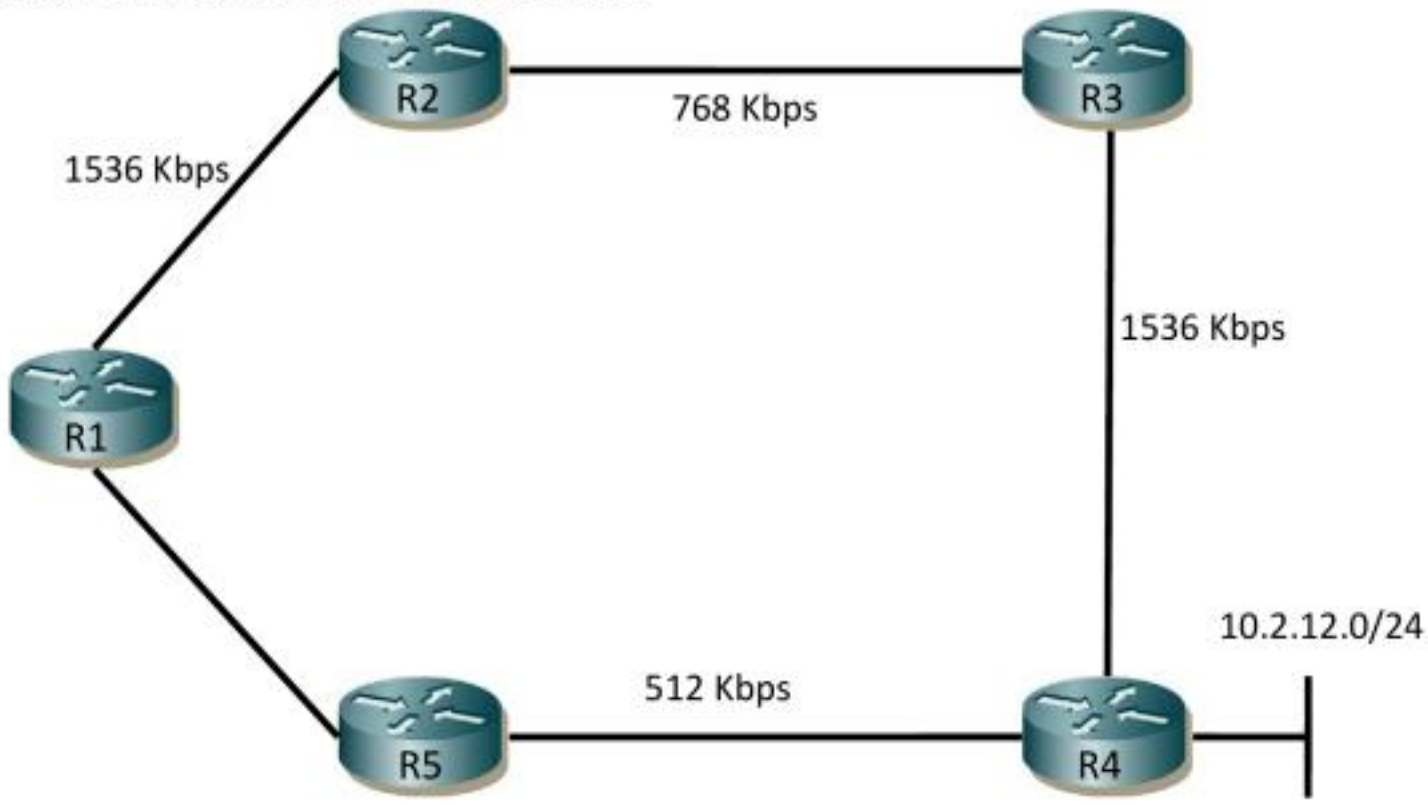

Figura 4-1: Calculul metricii

Calculăm costul căii de la R1 la reţeaua 10.2.12.0/24:
Bandwidth = 10^7 / min(bandwidth) = 10^7 / 768 = 13020
Delay = 3 x 20000 / 10 = 60000 / 10 = 6000
Metrica = 256 x (13020 + 6000) = 4869120
Calculăm costul căii de la R5 la reţeaua 10.2.12.0/24:
Bandwidth = 10^7 / min(bandwidth) = 10^7 / 512 = 19531
Delay = 3 x 20000 / 10 = 40000 / 10 = 4000
Metrica = 256 x (19531 + 4000) = 6023936

4.1.4 Maşina de stări finite DUAL

Diffusing Update Algorithm Finite State Machine (DUAL FSM) permite ruterului să identifice căile care nu conţin bucle. Pentru fiecare destinaţie sunt determinate toate căile fără bucle, ruta de cost minim va promova în tabela de rutare.

Un ruter ce primeşte un pachet de actualizare va extrage secvențial fiecare rută. Metrica unei rute din pachetul de actualizare va fi modificată în conformitate cu parametrii legăturii (interfeței) pe care a fost primită actualizarea. Rezultatul obținut va fi comparat cu restul informațiilor de rutare pentru respectiva destinație. Metrica cea mai mică pentru o destinație poartă numele de **distanță viabilă**.

Un **succesor** reprezintă un ruter adiacent prin care trece calea cu distanța viabilă minimă. Ruta de cost minim este introdusă în tabela de rutare. Implicit pot exista în tabela de rutare maximum 4 rute cu acelaşi cost minim, dar ruterul poate fi configurat pentru a accepta şase rute de cost minim pentru fiecare destinație.

Un **succesor viabil** este un ruter vecin prin care trece o cale care are o distanță viabilă mai mare decât a succesorului curent, dar care are o distanță raportată mai mică decât distanța viabilă a succesorului curent. Aceste rute nu vor fi introduse în tabela de rutare, dar vor fi păstrate în tabela de topologie (prezentată mai jos) ca rute de backup.

Dacă o rută care trece prin succesorul curent devine invalidă, sau dacă un vecin îşi schimbă metrica, DUAL verifică existența succesorilor viabili pentru acea destinație. Dacă există cel puțin un succesor viabil, nu se declanşează recalcularea rutei. Dacă nu există nici un succesor viabil, se recalculează ruta pentru a determina un nou succesor.

De exemplu, în topologia din figura 4-1, pe R1 algoritmul DUAL consideră două căi pentru destinația 10.2.12.0/24, una prin ruterul R2 şi una prin ruterul R5.

Este calculată distanța viabilă pentru cele două rute: pentru R2 este 4869120 iar pentru R5 este 6023936. Rezultă că R2 este succesorul pentru această destinație. Pentru a afla dacă R5 este un succesor viabil trebuie calculată distanța raportată a rutei prin R5:

Bandwidth = 10^7 / min(bandwidth) = 10^7 / 512 = 19531

Delay = 3 x 20000 / 10 = 20000 / 10 = 2000

Metrica = 256 x (19531 + 2000) = 5511936

Distanța raportată este mai mare decât distanța viabilă a succesorului, deci R5 nu poate fi un succesor viabil. Prin urmare, dacă ruta prin R2 devine invalidă, ruterul R1 trebuie să declanşeze recalcularea rutei.

4.1.5 Tabelele EIGRP

Protocolul EIGRP construieşte şi menține 3 tabele: **tabela cu vecinii, tabela de topologie şi tabela de rutare**. Tabela cu vecini conține informații despre ruterele adiacente şi este folosită pentru a verifica primirea confirmărilor de la toți vecinii. Tabela de topologie conține toate căile din rețea. Cele mai bune căi din tabela de topologie sunt introduse în tabela de rutare.

Când un ruter descoperă şi formează o adiacență cu un vecin nou, introduce în tabela cu vecini adresa şi interfața pe care vecinul poate fi accesat. Tabela cu vecini este asemănătoare tabelei de adiacență a protocoalelor de rutare bazate pe starea conexiunii, deoarece asigură comunicația bidirecțională între vecinii direct conectați. Un interval de *hold time* este anunțat odată cu pachetul *Hello*, fiind timpul în care vecinul este considerat accesibil. Dacă acest interval expiră fără să fi fost primit nici un pachet *Hello*, atunci *DUAL* va fi anunțat de schimbarea de topologie. Pentru fiecare protocol de nivel 3 (IPv4, IPv6, IPX şi AppleTalk) se construieşte o tabelă de vecini separată, iar informațiile despre vecini, rute şi costuri nu sunt partajate între protocoale.

Când un ruter stabileşte o legătură de adiacență cu un nou vecin, îi va trimite un pachet *Update* cu rutele pe care le cunoaşte. Noul vecin de asemenea îi va trimite rutele cunoscute de el. Aceste informații sunt introduse în tabela de topologie, care va conține toate rutele anunțate de către

ruterele adiacente. Este important de observat că vor fi anunțate doar rutele care apar în tabela de rutare, aceasta fiind o caracteristică specifică protocoalelor de rutare bazate pe vectori de distanță. În tabela de topologie este stocată distanța raportată și distanța viabilă pentru fiecare rută în parte. Tabela de topologie se modifică doar atunci când o rută direct conectată sau o interfață se schimbă, sau când un vecin anunță o modificare a unei rute. O destinație poate avea ataşată în tabelă fie o stare activă, fie o stare pasivă. O destinație se află în starea activă dacă în acel moment se recalculează ruta, iar în caz contrar destinația se află într-o stare pasivă. Recalcularea este efectuată atunci când ruta ce trece prin succesorul curent devine nevalidă și nu există un succesor viabil. Recalcularea debutează prin trimiterea pachetelor de tip *Query* (definite mai jos) la toți vecinii. Acest proces se va repeta în mod recursiv la toți vecinii care nu au o rută către destinație. Când ruterul primeşte răspuns de la toți vecinii va trece destinația în modul pasiv și va alege un succesor.

Ruterul compară toate distanțele viabile pentru o anumită destinație și selectează ruta cu cea mai mică distanță viabilă. Această rută de cost minim identifică și ruterul succesor, și va fi plasată in tabela de rutare. Distanța viabilă a acestei rute devine metrica EIGRP pentru destinația respectivă.

4.1.6 Formatul pachetelor EIGRP

Protocolul EIGRP foloseşte 5 tipuri de pachete pentru comunicarea între rutere: *Hello*, *Update*, *Ack*, *Query* şi *Reply*. Pachetele sunt încapsulate direct în IP.

Rolul pachetelelor *Hello* este de a identifica și verifica vecinii. Pachetele de tip *Hello* sunt trimise de către ruterele care rulează EIGRP la intervale fixe dar configurabile, numite intervale *Hello*. Aceste intervale depind de lățimea de bandă a interfeței. Pachetele sunt trimise prin transmisie multicast la adresa 224.0.0.10 și nu sunt confirmate de către destinație. Fiecare ruter stochează în tabela cu vecini informațiile din pachetele Hello primite de la vecinii săi. Dacă nici un pachet Hello nu este primit într-un interval de Hold (Hold time), ruterul consideră ca vecinul nu mai este accesibil. În mod implicit, intervalul Hold este de trei ori mai mare decat intervalul Hello, dar ambele pot fi configurate. Spre deosebire de OSPF, EIGRP nu necesită ca intervalele să fie egale pentru doi vecini care comunică, ci acestea își învață reciproc intervalele. Pentru o lățime de bandă de până la 1.544 Mbps, de exemplu în cazul Multipoint Frame Relay, intervalul Hello este de 60 de secunde iar intervalul Hold este de 180 de secunde. Pentru lățimi de bandă mai mari de 1.544 Mbps, de exemplu T1 sau Ethernet, intervalul Hello este de 5 secunde iar intervalul Hold este de 15 secunde.

Pachetele de tip *Update* sunt folosite pentru a anunța rutele. Atunci când un nou vecin este descoperit, ruterele trimit pachete Update prin transmisie unicast, pentru ca ruterul vecin să-şi construiască tabela de topologie. De asemenea, ele sunt trimise prin multicast atunci când se modifică o rută. Dacă se primeşte un Update care anunță că o cale nu mai există, ruterul își va întreba vecinii prin multicast dacă mai au o cale spre aceeași destinație. Dacă un anume vecin nu răspunde, îl va întreba din nou printr-un pachet unicast. Va renunța dupa ce a încercat de 16 ori fără să fi primit nici un răspuns. Pachetele de tip Update sunt trimise în mod sigur, deoarece necesită confirmare din partea destinației. Pachetele de tip *Query* sunt folosite atunci când un ruter are nevoie de informații specifice de la vecinii săi. Atunci când un ruter își pierde succesorul pentru o anumită destinație și nu are nici un succesor viabil, plasează ruta în starea activă și trimite un pachet de tip Query prin transmisie multicast către toți vecinii săi. Pachetele de tip Update sunt trimise în mod sigur.

Pachetele de tip *Reply* sunt folosite pentru a răspunde prin unicast la pachetele de tip Query. Răspunsul este că există o cale alternativă sau nu există nici o cale spre acea destinație. Pachetele de tip Reply sunt trimise în mod sigur.

Pachetele *Ack* sunt trimise pentru a confirma pachetele transmise în mod sigur, mai exact pachetele de tip Update, Query şi Reply. Pachetele de confirmare sunt trimise în mod unicast.

4.2 Concepte avansate

4.2.1 Sumarizarea rutelor

Sumarizarea rutelor determină o dimensiune mai mică a tabelei de rutare, iar procesul de actualizare a rutelor consumă mai puțină lățime de bandă. Sumarizarea automată a rutelor la granița clasei din care face parte rețeaua este configurată implicit pentru rutele învățate prin EIGRP. EIGRP nu sumarizează rețelele care nu participă la rutare.

În anumite cazuri sumarizarea automată nu este potrivită, de exemplu în cazul rețelelor discontinue. În aceste situații, sumarizarea automată trebuie dezactivată.

Pot fi configurate manual sumarizări de orice dimensiune în rețeaua internă, cât timp o rută mai specifică există în tabelă. Când ruta specifică va fi scoasă din tabela de rutare, ruta sumarizată va fi de asemenea scoasă.

4.2.2 Echilibrarea traficului

Echilibrarea traficului (*load balancing*) de cost egal este procesul prin care ruterul distribuie traficul către toate interfețele care au asociate rute cu metrici egale către aceeaşi destinație. Procesul de load balancing conduce la folosirea tuturor rutelor către aceaşi destinație şi creşte lățimea de bandă efectivă. EIGRP face în mod implicit echilibrarea traficului pentru toate căile de cost egal, plasând cel mult patru rute de cost egal în tabela de rutare. Ruterul poate fi însă configurat să accepte maximum şase rute de cost egal.

O caracteristică importantă a protocolului EIGRP este faptul că poate echilibra traficul pe mai multe rute de cost diferit, ținând cont de o valoare a variației acceptabile a metricilor folosite în procesul de *load balancing*. Mai precis, vor fi acceptate toate rutele cu metrica mai mică decât valoarea obținută prin înmulțirea valorii variației cu metrica rutei de cost minim. Aceste rute trebuie de asemenea să aibă distanța raportată mai mică decât distanța viabilă a succesorului. Prin urmare, procesul de load balancing poate folosi doar succesori şi succesori viabili. Această condiție previne buclele de rutare.

4.2.3 Scalabilitatea EIGRP în rețele mari

EIGRP este un protocol scalabil deoarece, odată cu creşterea în dimensiune a rețelei, performanța nu se diminuează şi procesul se ajustează rapid la schimbări. Protocolul oferă soluții pentru asigurarea scalabilității în rețele mari, de exemplu prin folosirea ruterelor *stub* pentru a limita domeniul în care se trimit pachete de tip Query.

Următorii factori pot afecta scalabilitatea unei rețele: cantitatea de informație schimbată între rutere, numărul de rutere, adâncimea topologiei şi numărul de căi alternative prin rețea. Cu cât cantitatea de informație este mai mare, cu atât ruterele trebuie să trimită mai multă informație atunci când apare un vecin nou sau atunci când sunt schimbări de topologie. Consumul de resurse pe un ruter este legat direct de numărul de rutere aflate în acea rețea.

Adâncimea unei topologii se referă la numărul de hopuri pe care informația trebuie să le parcurgă pentru a ajunge la toate ruterele din rețea. Latența propagării şi a procesului de investigare prin pachete Query încetinesc convergența rețelei cu cât adâncimea este mai mare. Este recomandată limita maximă de 7 hopuri.

O rețea ar trebui să ofere căi alternative pentru a nu exista vulnerabilitatea unui punct singular de eşec (*single point of failure*). Cu toate acestea, prea multe căi alternative pot duce la probleme de convergență, deoarece procesul de rutare trebuie să exploreze toate căile pentru rutele pierdute.

4.2.4 EIGRP Queries

În cazul în care o rută devine nevalidă şi nu există succesor viabil în tabela de topologie, va trebui găsită o cale alternativă pentru acea destinație. Ruta intră în starea activă şi ruterul trimite pachete de tip Query către toți vecinii săi, în afară de vecinul prin care trecea ruta declarată nevalidă, întrebând dacă au o rută către destinația specificată.

Dacă vecinul care a primit un Query are o rută alternativă, va trimite un pachet Reply conținând ruta. Dacă vecinul nu are nici o rută către destinație trimite pachete Query la toți vecinii săi întrebând de o cale alternativă. Astfel pachetele de tip Query se propagă prin rețea formând un arbore de cereri. Când un ruter are o rută alternativă opreşte propagarea pachetelor pentru ramura respectivă. Totuşi, pe celelalte ramuri propagarea nu este oprită.

Ruterul trebuie să primească toate răspunsurile de la vecinii săi pentru a putea calcula noul succesor. Abia după aceea va putea ieşi din starea activă şi intra din nou în starea pasivă. Dacă ruterul nu primeşte toate răspunsurile într-o perioadă maximă de 3 minute, va intra în starea *Stuck in Active* (*SIA*). Rutele aflate în SIA reprezintă una dintre probleme cele mai mari ale unei rețele EIGRP. O rută intrată în starea SIA determină resetarea legăturii cu vecinul care nu răspunde, ceea ce conduce la intrarea in starea activă a tuturor rutelor care treceau prin vecinul respectiv.

Motivele pentru care o rută poate intra în starea SIA sunt:

- ruterul vecin nu poate răspunde deoarece procesorul este folosit intensiv sau nu poate fi alocată memorie pentru a procesa cererea şi a livra un răspuns;
- legătura dintre ruter şi vecinul său are probleme şi pachetele Query şi Reply sunt pierdute;
- o defecțiune poate cauza funcționarea unidirecțională a legăturii;
- există prea multe căi alternative prin rețea, fapt ce determină probleme de convergență iar ruta intră în SIA deoarece aşteaptă răspunsurile pentru toate pachetele Query propagate pe toate căile alternative.

4.2.5 EIGRP Stub

Un ruter este *stub* dacă este conectat la un ruter de nivel central (*core*), dar traficul din core nu trece prin el.

Un ruter stub EIGRP îşi va anunța vecinii despre faptul că este stub iar vecinii vor înceta să-i mai trimită pachete de tip Query. În schimb, ruterul care are conectat un ruter stub va răspunde în locul acestuia. Cu toate acestea, ruterul stub va primi anunțuri cu rute exact ca un ruter normal.

Configurarea unui ruter ca stub va duce la creşterea scalabilității rețelei şi la minimizarea utilizării memoriei şi a procesorului ruterului respectiv, fiind o configurație potrivită pentru ruterele încete şi cu memorie puțină.

4.2.6 Autentificarea EIGRP

Pentru a garanta sursa informațiilor pe care le primeşte un ruter EIGRP, protocolul permite autentificarea vecinilor înainte de a schimba informații de rutare cu aceştia.

Protocolul EIGRP suportă două tipuri de autentificare: parole simple şi MD5. Parolele simple sunt trimise în clar şi potrivite cu cheia destinatarului. Ele nu oferă siguranța, deoarece oricine poate captura traficul are acces şi la această cheie.

Dat fiind că parolele simple nu sunt potrivite pentru securizarea comunicației, se recomandă folosirea parolelor criptate prin *MD5*. MD5 este o funcție criptografică de tip rezumat (*hash*) unidirecțională prin care se obține o valoare de lungime fixă de 128 de biți. Cheile criptate cu MD5 sunt sigure deoarece din traficul capturat nu se poate obține cheia inițială.

Folosind o cheie criptată prin MD5 în fiecare pachet, EIGRP previne acceptarea de către ruter a informațiilor false şi neautorizate. Pentru autentificarea prin MD5, un identificator de cheie şi o cheie de autenificare trebuie specificate pentru ambele rutere care comunică. Fiecare identificator de cheie are ataşată o cheie distinctă. Mai multe chei pot fi grupate într-un lanț de chei (*key chain*). Pentru fiecare identificator de cheie se poate specifica un interval de timp în care cheia este validă. Când se primeşte un pachet de autentificare, se parcurge lanțul de chei de la identificatorul cel mai mic la cel mai mare pentru a se identifica cheia potrivită.

4.3 Configurarea protocolului

4.3.1 Configurări de bază

Pentru a configura protocolul de rutare EIGRP este necesară parcurgerea următorilor paşi:

Pas 1. Activarea procesului EIGRP şi specificarea unui număr de sistem autonom. Cel mai adesea el nu are o legătură directă cu numărul sistemului autonom din care face parte ruterul, sigura constrângere fiind ca aceaşi valoare să fie configurată pe fiecare dintre ruterele din cadrul rețelei EIGRP.

```
Router(config)#router eigrp numar_AS
```

Pas 2. Specificarea rețelelor care fac parte din sistemul autonom. Această comandă indică ce interfețe participă la rutarea prin EIGRP şi ce rețele sunt anunțate de către ruter.

```
Router(config-router)#network retea [masca_wildcard]
```

Masca poate fi specificată pentru a identifica anumite subrețele ale unei clase de rețea şi poate fi introdusă în formatul invers sau în formatul normal de mască de rețea.

Pas 3. Poate fi definită lățimea de bandă a unei legături. Această valoare configurată nu afectează viteza actuală a legăturii, dar este folosită pentru calculele efectuate de protocolul de rutare. Folosind următoarea comandă, lățimea de bandă este specificată în Kbps:

```
Router(config-if)#bandwidth valoare
```

Dacă nu este specificată în mod explicit lățimea de bandă pentru o legatură serială, protocolul va presupune că lățimea de bandă este cea pentru o legătura T1. Dacă legătura fizică este mai înceată, ruterul poate să nu mai conveargă sau actualizările pot fi pierdute.

Pentru exemplificarea folosirii acestor comenzi, se va considera următorul exemplu de configurare. Se va folosi topologia din figura 4-2. Cele 4 rutere trebuie configurate ca aparținând aceluiaşi sistem autonom cu numărul 101. Rețeaua 172.16.9.0/24 nu trebuie să facă parte din rețeaua EIGP.

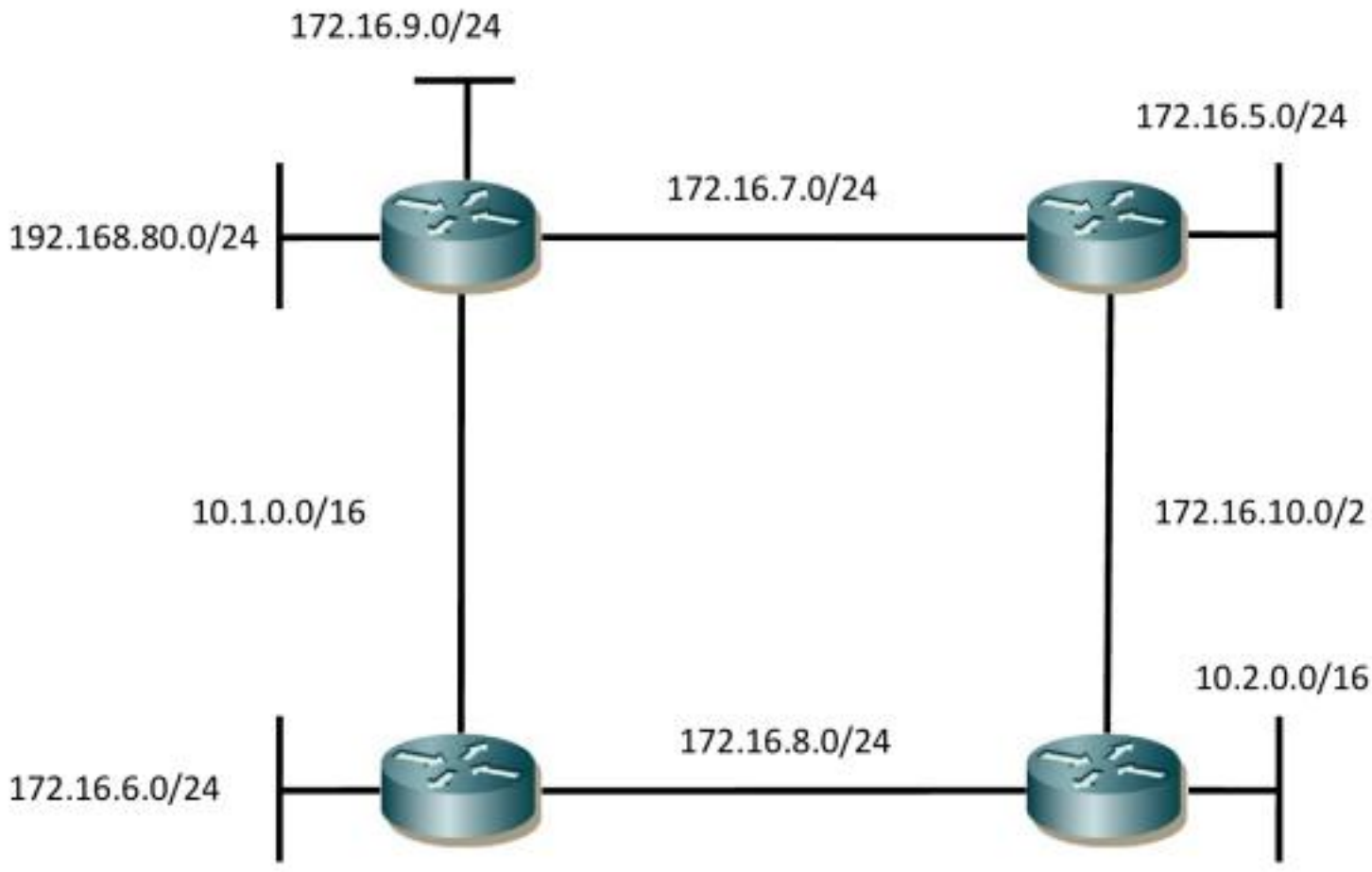

Figura 4-2: Exemplu de configurare

Ruterul R1 va trebui configurat în felul următor:

```
R1(config)#router eigrp 101
R1(config-router)#network 192.168.80.0 0.0.0.255
R1(config-router)#network 172.16.7.0 0.0.0.255
R1(config-router)#network 10.1.0.0 0.0.255.255
```

Pentru ruterul R2 vor trebui introduse umătoarele comenzi:

```
R2(config)#router eigrp 101
R2(config-router)#network 172.16.5.0 0.0.0.255
R2(config-router)#network 172.16.7.0 0.0.0.255
R2(config-router)#network 172.16.10.0 0.0.0.255
```

Pentru ruterul R3 următoarele comenzi:

```
R3(config)#router eigrp 101
R3(config-router)#network 10.2.0.0 0.0.255.255
R3(config-router)#network 172.16.10.0 0.0.0.255
R3(config-router)#network 172.16.8.0 0.0.0.255
```

Pentru ruterul R4 următoarele comenzi:

```
R3(config)#router eigrp 101
R3(config-router)#network 172.16.6.0 0.0.0.255
R3(config-router)#network 10.1.0.0 0.0.255.255
R3(config-router)#network 172.16.8.0 0.0.0.255
```

S-au specificat doar rețelele care sunt direct conectate. Măşti de tip wildcard au fost folosite pentru a indica subrețelele adresei de clasa B 172.16.0.0. Dacă nu s-ar fi specificat măştile ruterul ar fi stocat adresele rețelelor sumarizate la nivel de clasă. De exemplu R1 ar fi stocat rețelele 192.168.80.0, 172.16.0.0 şi 10.0.0.0. În această situaţie rețeaua 172.16.9.0/24 ar fi fost inclusă automat în rețeaua EIGRP.

4.3.2 Propagarea rutei implicite

Se poate crea o rută implicită EIGRP folosind următoarea comandă:

```
Router(config)#ip default-network retea
```

Ruterul pe care a fost configurată o rută implicită va include în anunțul său această rută. Ruterele din rețea vor folosi ruterul *next-hop* către acea rețea pentru a introduce în tabela de rutare ruta implicită. De aceea rețeaua destinație trebuie să fie accesibilă de pe ruterul inițial pentru ca să poată fi anunțată către celelalte rutere. De asemenea, destinația trebuie sa fie cunoscută de ruterele din rețea pentru a putea fi stocată ruta implicită. În concluzie rețeaua destinație trebuie să fie declarată în procesul EIGRP sau să existe o rută statică redistribuită în EIGRP.

Mai multe rute implicite pot fi configurate, iar ruterele care primesc anunțurile vor folosi metrica EIGRP pentru a decide care este cea mai bună rută.

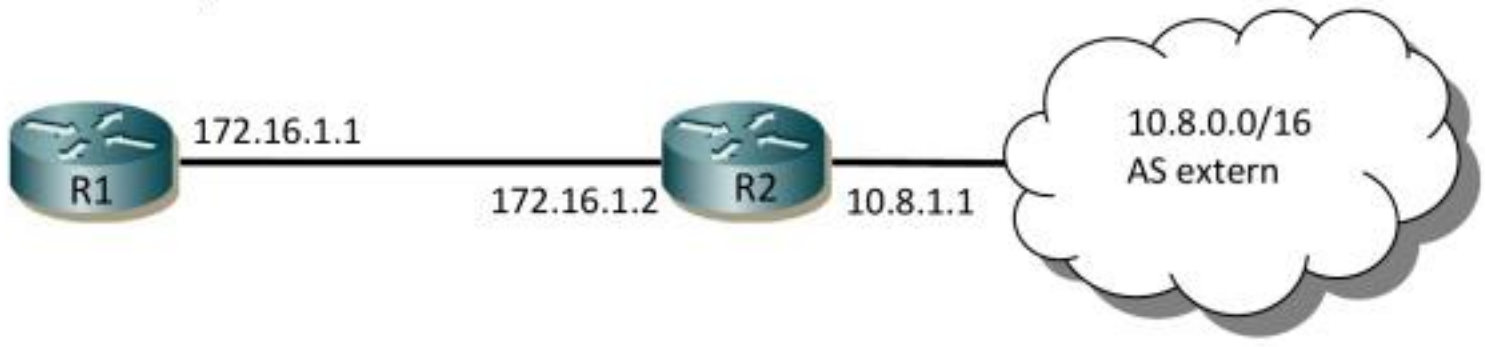

Figura 4-3: Figura 4-4: Propagarea rutei implicite

Folosind topologia din figura 4-3, se va configura o rută implicită pe R2, şi se va observa propagarea ei pe ruterul R1. Configurația care trebuie efectuată pe ruterul R2 este:

```
R2(config)#ip default-network 10.8.0.0
R2(config)#router eigrp 102
R2(config-router)#network 172.16.1.0 0.0.0.255
R2(config-router)#network 10.8.0.0 0.0.255.255
```

Folosind comanda **`ip default-network 10.8.0.0`**, pe ruterul R1 a fost configurată rețeaua 10.8.0.0/16 ca fiind candidată pentru rolul de rețea implicită. Comanda va genera în mod automat următoarea rută statică: **`ip route 10.8.0.0 255.255.0.0 10.8.1.1`**.

După cum se observă, rețeaua externă 10.8.0.0/16 a fost declarată în procesul EIGRP, deci va fi cunoscută de către ruterele din sistemul autonom 102.

Observăm tabela de rutare pe ruterul R2:

```
R2#show ip route
Gateway of last resort is not set
C       172.16.1.0/24 is directly connected, Ethernet0/0
S*      10.8.0.0/16 [1/0] via 10.8.1.1
```

Ruta implicită este învățată prin EIGRP de către ruterul R1 care va avea următoarea tabelă de rutare:

```
R1#show ip route
Gateway of last resort is 172.16.1.2 to network 10.8.0.0
C       172.16.1.0/24 is directly connected, Ethernet0/0
D*      10.8.0.0/16 [90/10486122] via 172.16.1.2 00:00:10 Ethernet0/0
```

Se poate observa că în tabela de rutare se va instala o rută implicită către rețeaua 10.8.0.0/16 care poate fi accesată de pe ruterul R1 prin adresa next-hop 172.16.1.2 care aparține ruterului R2.

4.3.3 Verificarea funcționării EIGRP

Se va folosi topologia din figura 4-5.

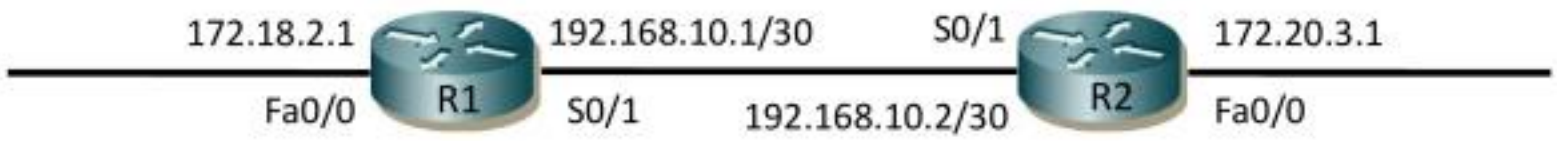

Figura 4-5: Verificarea funcționării EIGRP

Configurația de pe ruterul R1 este:

```
R1(config)#router eigrp 103
R1(config-router)#network 172.18.2.0 0.0.0.255
R1(config-router)#network 192.168.10.0
```

Iar configurația de pe ruterul R2 este:

```
R2(config)#router eigrp 103
R2(config-router)#network 172.20.3.0 0.0.0.255
R2(config-router)#network 192.168.10.0
```

Tabela cu vecinii se poate poate vizualiza folosind următoarea comandă:

```
R1#show ip eigrp neighbors
IP-EIGRP neighbors for process 103
H  Address            Interface   Hold  Uptime      SRTT  RTO   Q
   Seq
                                  (sec)             (ms)        Cnt  Num
0  192.168.10.2       Se0/1       12    00:10:20    17    2280  0     14
```

Coloanele tabelului au semnificația următoare:

- **H** este un număr folosit pentru a identifica vecinii, şi a stoca ordinea în care au fost învățați;
- **Address** este adresa IP a vecinului;
- **Interface** este interfața de pe ruterul local pe care poate fi accesat vecinul;
- **Hold** este intervalul maxim în care ruterul aşteaptă un răspuns de la vecin până să considere legătura nevalidă. Primirea unui pachet de tip Hello resetează acest interval;
- **Uptime** este timpul scurs de la adăugarea vecinului;
- **SRTT (smoothed round-trip time)** este timpul mediu în care un pachet EIGRP ajunge la vecin şi un packet Ack este primit de ruterul local;
- **RTO (retransmission timeout)** este intervalul de timp în care ruterul aşteaptă un pachet de confirmare de la vecin până să retransmită un pachet trimis în mod sigur;
- **Q Cnt (queue count)** este numărul de pachete care se află în coadă aşteptând să fie trimise vecinului. Dacă numărul este mai mare decât 0, este posibil să fie o congestie;
- **Seq Num** este numărul de secvență a ultimului pachet de tip Update, Query sau Reply primit de la vecin.

Tabela de rutare va fi inspectată folosind comanda **`show ip route`**:

```
R1#show ip route
Codes: C - connected, S - static, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2

Gateway of last resort is not set

D       172.20.0.0/16 [90/40514560] via 192.168.10.2, 00:07:09, Serial0/1
     172.18.0.0/16 is variably subnetted, 2 subnets, 2 masks
D               172.18.0.0/16 is a summary, 00:05:31, Null0
C               172.18.2.0/24 is directly connected, FastEthernet0/0
          192.168.10.0/24 is variably subnetted, 2 subnets, 2 masks
C               192.168.10.0/30 is directly connected, Serial0/1
D               192.168.10.0/24 is a summary, 00:05:31, Null0
```

Iar pentru a obţine doar rutele EIGRP din tabela de rutare se foloseşte următoarea comandă:

```
R1#show ip route eigrp
D       172.20.3.0/16 [90/40514560] via 192.168.10.2, 00:07:09, Serial0/1
     172.18.0.0/16 is variably subnetted, 2 subnets, 2 masks
D                172.18.0.0/16 is a summary, 00:05:31, Null0
           192.168.10.0/24 is variably subnetted, 2 subnets, 2 masks
D                192.168.10.0/24 is a summary, 00:05:31, Null0
```

EIGRP va introduce în tabela de rutare 3 tipuri de rute: interne, externe şi sumarizate. Cele interne vor fi reprezentate prin litera D, cele externe prin literele D EX.

După adresa reţelei, între paranteze pătrate se află doua numere. Primul număr reprezintă distanţa administrativă care pentru rute EIGRP interne este 90 şi pentru rute EIGRP externe este 170. Al doilea număr reprezintă metrica rutei care este egală cu distanţa viabilă din tabela de topologie.

Câmpul următor va specifica adresa ruterului next-hop către destinaţia respectivă. Această adresă este egală cu adresa succesorului din tabela de topologie.

Urmează un câmp care reprezintă timpul scurs de la ultimul anunţ despre acestă rută. În EIGRP rutele nu sunt trimise periodic, doar atunci când se schimbă adiacenţele.

Ultimul câmp este interfaţa pe care pachetele pentru acea destinaţie trebuie trimise. Rutele sumarizate în mod automat au asociată interfaţa Null0 care este o interfaţă software. Această configuraţie previne rutarea pachetelor destinate unor subreţele care fac parte din ruta sumarizată, dar care nu există în realitate şi nu există rută specifică pentru ele.

Comanda **`show ip protocols`** oferă informaţii despre toate protocoalele de rutare dinamice care rulează pe ruter:

```
R1#show ip protocols
Routing Protocol is "eigrp 103"
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Default networks flagged in outgoing updates
  Default networks accepted from incoming updates
  EIGRP metric weight K1=1, K2=0, K3=1, K4=0, K5=0
  EIGRP maximum hopcount 100
  EIGRP maximum metric variance 1
  Redistributing: eigrp 100
  EIGRP NFS-aware route hold timer is 240s
  Automatic network summarization is in effect
  Automatic address summarization:
    192.168.10.0/24 for Serial0/1
      Summarizating with metric 40512000
    172.18.0.0/16 for FastEthernet0/0
      Summarizating with metric 28160
  Maximum path: 4
  Routing for Networks:
    172.18.2.0/24
    192.168.10.0
  Routing Information Sources:
    Gateway         Distance      Last Update
    (this router)         90      00:01:08
    192.168.10.2          90      00:01:08
  Distance: internal 90 external 170
```

Folosind comanda anterioară pot fi vizualizaţi parametri K, numărul de hopuri şi variaţia. Ruterele trebuie să aibă aceleaşi valori K pentru a putea stabili o adiacenţă.

Se poate observa că sumarizarea automată este activată şi faptul ca ruterul poate face *load balancing* pentru patru rute de cost egal.

De asemenea, se poate determina pentru ce rețele sunt rutate pachete. Dacă atunci când a fost introdusă comanda **network** s-a specificat un *wildcard mask*, rețeaua este reprezentată cu prefix, altfel este stocată rețeaua la nivel de clasă.

Secțiunea **Routing Information Sources** identifică ruterele cu care s-a stabilit o adiacență, incluzând şi ruterul local.

Sunt precizate şi distanțele administrative pentru rutele EIGRP interne şi externe.

Comanda **show ip eigrp interfaces** oferă informații despre interfețele configurate pentru EIGRP:

```
R1#show ip eigrp interfaces
IP-EIGRP interfaces for process 103

                Xmit Queue   Mean   Pacing Time   Multicast    Pending
Int     Peers   Un/Reliable  SRTT   Un/Reliable   Flow Timer   Routes
Fa0/0     0        0/0         0        0/1           0           0
Se0/1     1        0/0        10       10/380       424          0
```

Coloanele tabelului au semnificația următoare:

- **Interface** este interfața pe care este configurat EIGRP;
- **Peers** este numarul de vecini direct conectați pe acea interfață;
- **Xmit Queue Un/Reliable** este numărul de pachete aflate în cozile de transmisie;
- **Mean SRTT** este valoarea medie a parametrului SRTT;
- **Pacing Time Un/Reliable** este timpul folosit pentru a determina când trebuie trimis un pachet;
- **Multicast Flow Timer** este numărul maxim de secunde în care ruterul trimite pachete EIGRP de tip multicast;
- **Pending Routes** este numărul de rute folosite pentru pachetele care se află în cozile de transmisie.

Tabela de topologie poate fi vizualizată folosind comanda:

```
R1#show ip eigrp topology
IP-EIGRP Topology Table for AS(103)/ID(192.168.10.1)

Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
       r - reply Status, s - sia Status

P 192.168.10.0/30, 1 successors, FD is 40512000
        via Connected, Serial0/1
P 192.168.10.0/24, 1 successors, FD is 40512000
        via Summary (40512000/0), Null0
P 172.18.0.0/16, 1 successors, FD is 28160
        via Summary (40512000/0), Null0
P 172.18.2.0/24, 1 successors, FD is 28160
        via Connected, FastEthernet0/0
P 172.20.0.0/16, 1 successors, FD is 40514560
        via 192.168.10.2 (40514560/28160), Serial0/1
```

Din comanda anterioară se poate observa că sistemul autonom este 103 iar identificatorul ruterului R1 este 192.168.10.1. Identificatorul se alege după următorul algoritm: dacă există adrese configurate pe interfețele de loopback, se va alege cea mai mare adresă, iar dacă nu sunt configurate adrese pe interfețele de loopback, se va alege cea mai mare adresă dintre interfețele configurate.

Prima coloană reprezintă starea rutei respective, care poate fi:

- **P (Passive)** – rețeaua este accesibilă, ruta se poate instala în tabela de rutare;
- **A (Active)** – rețeaua nu este accesibilă, ruta nu poate fi instalată în tabela de rutare. Există pachete de tip Query trimise pentru această rută;

- **U (Update)** – ruterul a primit un pachet de tip Update cu această rețea sau ruterul așteaptă un pachet Ack pentru un Update;
- **Q (Query)** – ruterul a trimis un pachet Query pentru această rețea sau ruterul așteaptă un pachet Ack pentru un Query;
- **R (Reply status)** – ruterul generează un pachet Reply pentru acestă rețea sau așteaptă un packet Ack pentru un Reply;
- **S (Stuck-in-active status)** – ruta este în starea SIA. Există probleme de convergență.

Pentru fiecare destinație este indicat numărul de succesori, distanța viabilă (FD), adresa ruterului next hop urmată de un set de două numere reprezentând distanța viabilă și distanța raportată.

Pentru a obține numărul de pachete EIGRP primite și trimise se folosește comanda:

```
R1# show ip eigrp traffic

IP-EIGRP Traffic Statistics for AS 103
  Hellos sent/received: 218/205
  Updates sent/received: 7/23
  Queries sent/received: 2/0
  Replies sent/received: 0/2
  Acks sent/received: 21/14
```

Se observă că au fost trimise 218 pachete și au fost primite 205 pachete de tip Hello.

4.3.4 Configurarea sumarizării

După cum s-a precizat în subcapitolul 4.2.1, protocolul EIGRP realizează sumarizarea la granița clasei în mod implicit.

Revenind la figura 4-2, ruterul R4 are conectată rețeaua 10.1.0.0/16 iar ruterul R3 are conectată rețeaua 10.2.0.0/16. Dacă sumarizarea automată este activată, ruterul R4 va anunța ruterul R3 că are o rută către rețeaua 10.0.0.0/8, iar ruterul R3, pentru că se consideră direct conectat la 10.0.0.0/8, va ignora anunțul. În această situație trebuie dezactivată sumarizarea automată.

Următoarea comandă dezactivează sumarizarea automată pentru procesul EIGRP:

```
Router(config-router)#no auto-summary
```

Pentru a anunța că un anumit interval de adrese se poate accesa printr-o anumită interfață se folosește comanda:

```
Router(config-if)#ip summary-address eigrp numar_AS adresa_IP masca
```

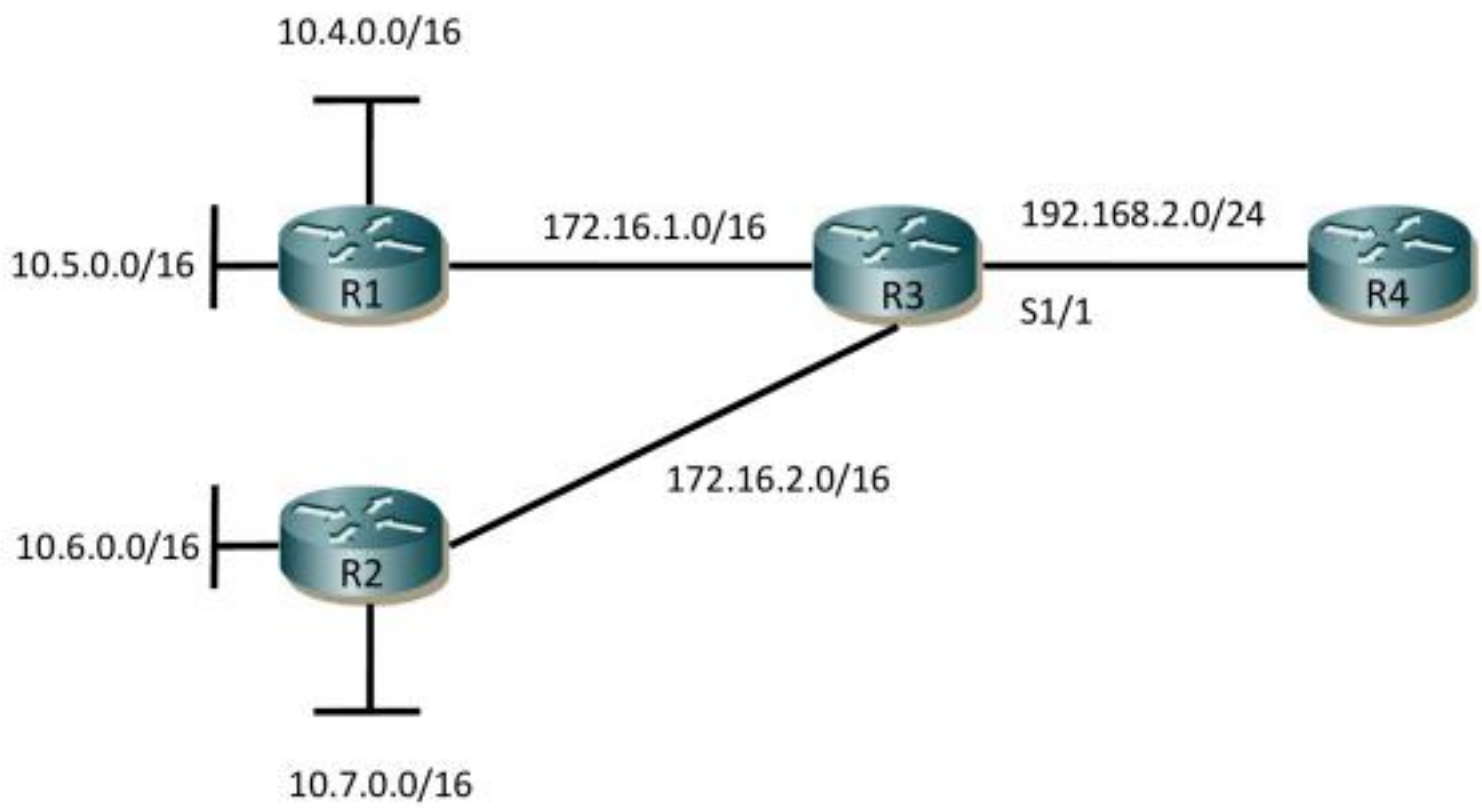

Figura 4-6: Sumarizare manuală

Folosind topologia din figura 4-5, o exemplificare a sumarizării manuale este următoarea:

```
R3(config)#interface s1/1
R3(config-if)#ip summary-address eigrp 103 10.4.0.0 255.255.252.0
```

Comanda anterioară determină ca rețeaua 10.4.0.0/14 să fie accesibilă de pe ruterul R4 prin interfața s1/1 a ruterului R3.

Pe ruterul R1 şi R2 trebuie dezactivată sumarizarea automată:

```
R1(config)#router eigrp 103
R1(config-router)#no auto-summary
R2(config)#router eigrp 103
R2(config-router)#no auto-summary
```

4.3.5 Configurarea balansării traficului

Procesul de load balancing a fost prezentat în subcapitolul 4.2.2. Pe un ruter se pot configura numărul maxim de rute de cost egal şi variația metricii rutelor de cost diferit.

Configurarea numărului maxim de rute de cost egal care pot fi introduse în tabela de rutare se face prin comanda următoare:

```
Router(config-router)#maximum-paths maxim
```

Această configurație va duce la realizarea de load balancing pe rutele de cost egal din tabela de rutare.

De exemplu, pentru a accepta 6 rute de cost egal, ruterul trebuie configurat în modul următor:

```
Router(config-router)#maximum-paths 6
```

Pentru a dezactiva load balancing se va preciza valoarea maximă ca fiind 1.

Gradul de echilibrare este stabilit prin următoarea comandă:

```
Router(config-router)#variance multiplicator
```

Această configurare permite introducerea în tabela de rutare a rutelor cu metrica mai mică decât valoarea obținută înmulțind multiplicatorul cu metrica cea mai mică pentru acea destinație.

Multiplicatorul poate lua o valoare între 1 si 128. Valoarea implicită este 1, ceea ce permite doar load balancing pentru rutele de cost egal.

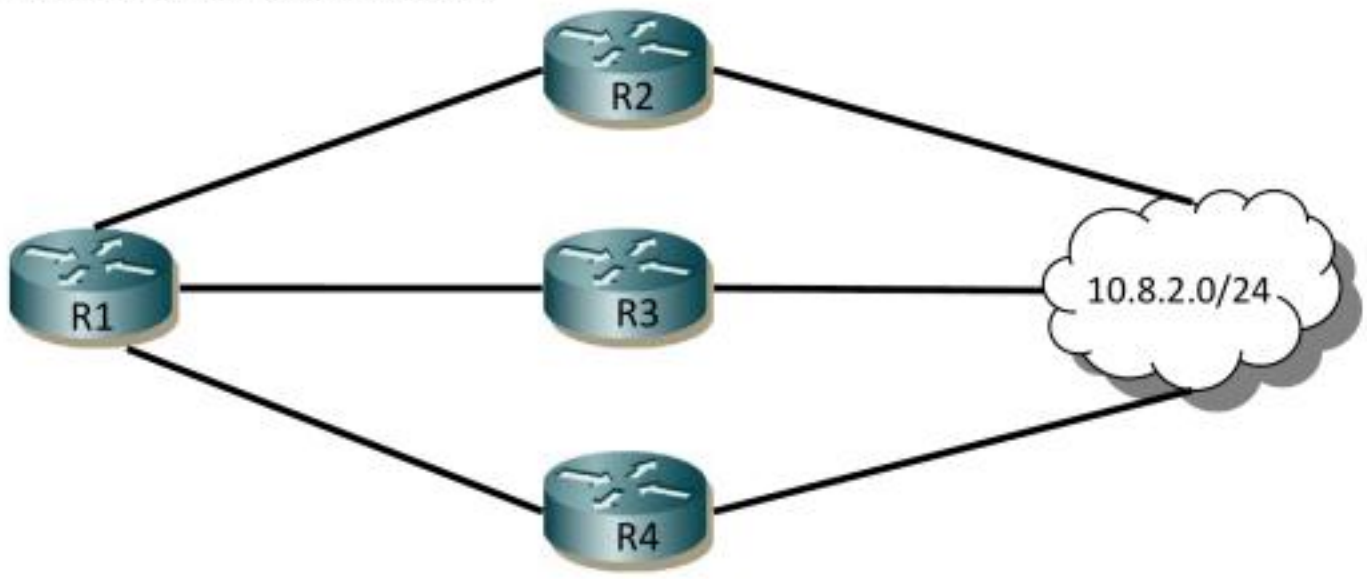

Figura 4-7: Load balancing cu costuri inegale

Considerăm topologia din figura 4-6. Tabela de topologie conține următoarele rute către destinația 10.8.2.0/24:

Vecin	Distanță viabilă	Distanță raportată
R2	30	15
R3	20	10
R4	40	30

Figura 4-8: Conținutul tabelei de topologie

Ruta succesor va trece prin R3 deoarece are cea mai mică distanță viabilă. Succesorul viabil este R1, deoarece are distanța raportată mai mică decât distanța viabilă a succesorului. R4 nu este un succesor viabil.

Inițial doar ruta prin R2 este introdusă în tabela de rutare. Pentru ca ruta prin R1 sa fie introdusă în tabela de rutare şi ruterul sa facă load balancing între cele două rute, trebuie să se configureze o valoare a variației care, înmulțită cu distanța viabilă a succesorului, să dea o valoare mai mare decât distanța viabilă a rutei prin R1. Acestă valoare este 2, deoarece 2 x 20 > 30. Ruta prin R4 nu va putea fi introdusă în tabela de rutare, chiar dacă configurăm variația ca fiind 3, deoarece R4 nu este un succesor viabil.

Se vor efectua următoarele configurări:

```
R1(config)#router eigrp 104
R1(config-router)#variance 2
```

4.3.6 Configurarea ruterului stub

Noțiunea de ruter stub a fost definită în subcapitolul 4.2.5. Pentru a configura un ruter ca fiind stub, se va folosi următoarea comandă:

```
Router(config-router)#eigrp stub [receive-only | connected | static |
summary]
```

Pentru a configura un ruter stub să trimită rutele direct conectate se va folosi argumentul *connected*. Această opțiune este activată în mod implicit. Dacă rutele direct conectate nu sunt configurate prin comanda **network**, vor trebui redistribuite în EIGRP prin comanda **redistribute connected**.

Argumentul *static* poate fi folosit pentru a determina ruterul să trimită rutele statice care au fost redistribuite în protocolul EIGRP prin comanda **`redistribute static`**.

Pentru ca un ruter stub să trimită rute sumarizate trebuie folosit argumentul *summary*. Vor fi trimise atât rutele sumarizate în mod automat, dar şi cele sumarizate folosind comanda **`ip summary-address`**.

Comanda **`eigrp stub`** folosită cu argumentul *receive-only* va împiedica ruterul sa trimită orice rută cunoscută către alt ruter din rețea. Dacă există o singură interfață pe ruter, este recomandată folosirea acestei comenzi, deoarece ruterul nu are alte rute decât cele primite pe legătura cu restul rețelei. Argumentul receive-only nu se poate folosi împreună cu celelalte argumente.

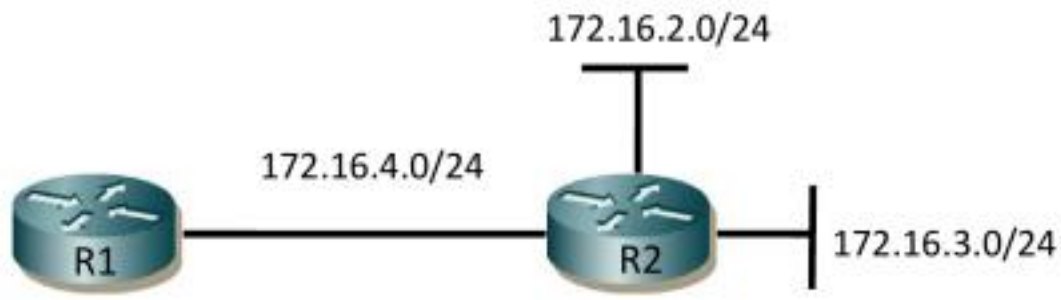

Figura 4-9: EIGRP Stub

Considerăm topologia din figura 4-7 şi următoarea configurație:

```
R2(config)#interface s0/0
R2(config-if)#ip sumary-address eigrp 105 172.16.2.0 255.255.254.0
R2(config)#router eigrp 105
R2(config-router)#network 172.16.4.0 0.0.0.255
R2(config-router)#network 172.16.2.0 0.0.0.255
```

Se poate observa că rețeaua 172.16.3.0/24 nu a fost specificată procesului EIGRP, dar este creată o rută sumarizată care conține şi rețeaua respectivă.

Dacă se va introduce comanda următoare, se va anunța către ruterul R1 doar rețeaua 172.16.2.0/24, deoarece ea a fost specificată procesului EIGRP.

```
R2(config-router)#eigrp stub connected
```

Comanda următoare va determina ruterul R2 să nu trimită nici o rută, deoarece nu a fost configurată şi redistribuită nici o rută statică.

```
R2(config-router)#eigrp stub static
```

Ruterul R2 va trimite o rută sumarizată 172.16.2.0/23 către R1:

```
R2(config-router)#eigrp stub summary
```

Comanda următoare va opri orice anunț către ruterul R1:

```
R2(config-router)#eigrp stub receive-only
```

4.3.7 Configurarea autentificării

Conceptele teoretice legate de autentificarea în cadrul protocolului EIGRP au fost prezentate în subcapitolul 4.2.6. Pentru a configura autentificarea prin MD5 pentru EIGRP trebuie parcurşi următorii paşi:

Pas 1. Intrarea în modul de configurare a interfeței pentru care se activează autentificarea.

Pas 2. Configurarea autentificării prin MD5 pentru EIGRP.

```
Router(config-if)#ip authentication mode eigrp numar_AS md5
```

Pas 3. Specificarea lanțului de chei folosit în autentificare

```
Router(config-if)#ip authentication key-chain eigrp numar_AS nume_lant
```

Pas 4. Intrarea în modul de configurare al cheilor.

```
Router(config)#key chain nume_lant
```

Pas 5. Specificarea identificatorului pentru cheie.

```
Router(config-keychain)#key identificator
```

Pas 6. Specificarea cheii (parolei).

```
Router(config-keychain-key)#key-string parola
```

Pas 7. Opțional, se poate specifica o perioadă de timp în care parola să fie valabilă pentru pachetele primite.

```
Router(config-keychain-key)#accept-lifetime timp_inceput [infinite |
timp_sfarsit | duration secunde]
```

Pas 8. Opțional, se poate specifica o perioadă de timp în care parola să fie folosită pentru pachetele trimise.

```
Router(config-keychain-key)#send-lifetime timp_inceput [infinite |
timp_sfarsit | duration secunde]
```

Figura 4-10: Autentificare prin MD5

Considerăm topologia din figura 4-8. Pachetele EIGRP între cele două rutere trebuie să fie criptate prin MD5.

Ruterul R1 trebuie configurat în modul următor:

```
R1(config)#key chain lant1
R1(config-keychain)#key 1
R1(config-keychain-key)#key-string cheie1
R1(config-keychain-key)#accept-lifetime 00:00:00 Oct 1 2009 infinite
R1(config-keychain-key)#send-lifetime 00:00:00 Oct 1 2009 00:10:00 Oct 1
2009
R1(config-keychain)#key 2
R1(config-keychain-key)#key-string cheie2
R1(config-keychain-key)#accept-lifetime 00:00:00 Oct 1 2009 infinite
R1(config-keychain-key)#send-lifetime 00:00:00 Oct 1 2009 infinite

R1(config)#interface s0/0
R1(config-if)#ip authentication mode eigrp 106 md5
R1(config-if)#ip authentication key-chain eigrp 106 lant1

R1(config)#router eigrp 106
R1(config-router)#network 172.18.2.0 0.0.0.255
R1(config-router)#network 192.168.10.0 0.0.0.255
```

Ruterul R1 va accepta pachetele EIGRP cu identificatorul cheii egal cu 1 sau cu 2 şi le va verifica valoarea MD5. Orice alte pachete vor fi ignorate. De asemenea ruterul R1 va trimite pachete

folosind cheia 1 timp de 10 minute în intervalul specificat, iar după expirarea intervalului pentru cheia 1, va folosi exclusiv cheia 2.

Ruterul R2 trebuie configurat în modul următor:

```
R2(config)#key chain lant2
R2(config-keychain)#key 1
R2(config-keychain-key)#key-string cheie1
R2(config-keychain-key)#accept-lifetime 00:00:00 Oct 1 2009 infinite
R2(config-keychain-key)#send-lifetime 00:00:00 Oct 1 2009 infinite
R2(config-keychain)#key 2
R2(config-keychain-key)#key-string cheie2
R2(config-keychain-key)#accept-lifetime 00:00:00 Oct 1 2009 infinite
R2(config-keychain-key)#send-lifetime 00:00:00 Oct 1 2009 infinite

R2(config)#interface s0/0
R2(config-if)#ip authentication mode eigrp 106 md5
R2(config-if)#ip authentication key-chain eigrp 106 lant2

R2(config)#router eigrp 106
R2(config-router)#network 172.20.3.0 0.0.0.255
R2(config-router)#network 192.168.10.0 0.0.0.255
```

Ruterul R1 va accepta pachetele EIGRP cu identificatorul cheii egal cu 1 sau cu 2 şi le va verifica valoarea MD5. Orice alte pachete vor fi ignorate. De asemenea ruterul R1 va trimite pachete folosind doar cheia 1 pentru că este prima în listă şi nu poate expira, prin urmare, cheia 2 nu va fi folosită niciodata pentru a trimite pachete.

În cazul în care cheile nu se potrivesc, nu se poate forma o adiacență între cei doi vecini.

Pentru a verifica dacă s-a format o adiacență se verifică tabela cu vecini:

```
R1#show ip eigrp neighbors
IP-EIGRP neighbors for process 105
H  Address            Interface   Hold  Uptime      SRTT  RTO   Q
   Seq
                                   (sec)             (ms)        Cnt
   Num
0  192.168.10.2       Se0/0       12    00:10:20    17    2280  0     14
```

De asemenea, atunci când s-a format o nouă adiacență se va afişa mesajul:

```
R1#
*Oct 1 00:00:59.503 GMT: %DUAL-5-NBRCHANGE: IP-EIGRP(0) 106: Neighbor
192.168.10.2 (Serial0/0) is up: new adjacency
```

Pentru depanarea autentificării prin MD5 se poate folosi comanda **debug eigrp packets**, care va afişa informații legate de pachetele EIGRP primite şi trimise:

```
R2#debug eigrp packets
EIGRP Packets debugging is on
    (UPDATE, REQUEST, QUERY, REPLY, HELLO, IPXSAP, PROBE, ACK, STUB,
SIAQUERY,SIAREPLY)
R2#
*Oct 1 16:38:38.321: EIGRP: received packet with MD5 authentication,
key id = 2
*Oct 1 16:38:38.321: EIGRP: Received HELLO on Serial0/0 nbr
192.168.10.2
*Oct 1 16:38:38.321:   AS 106, Flags 0x0, Seq 0/0 idbQ 0/0 iidbQ
un/rely 0/0 peerQ un/rely 0/0
```

Se observă că ruterul R2 primeşte de la R1 pachete EIGRP cu autentificare prin MD5, cu un identificator de cheie 2, iar autentificarea are loc cu succes.

Se fac următoarele configurări pe ruterul R2:

```
R2(config)#key chain lant2
```

```
R2(config-keychain)#key 2
R2(config-keychain-key)#key-string altacheie
```

Se observă că autentificarea pachetelor primite de la ruterul R1 nu mai poate avea loc deoarece ruterul R1 va trimite identificatorul de cheie 2 (cheia 1 a expirat după primele 10 minute), dar parolele configurate pe cele două rutere sunt diferite:

```
R2#debug eigrp packets
EIGRP Packets debugging is on
     (UPDATE, REQUEST, QUERY, REPLY, HELLO, IPXSAP, PROBE, ACK, STUB,
SIAQUERY, SIAREPLY)
R2#
*Oct 1 16:50:18.749: EIGRP: pkt key id = 2, authentication mismatch
*Oct 1 16:50:18.749: EIGRP: Serial0/0: ignored packet from
192.168.10.1, opcode = 5 (invalid authentication)
*Oct 1 16:50:18.749: EIGRP: Dropping peer, invalid authentication
*Oct 1 16:50:18.749: EIGRP: Sending HELLO on Serial0/0
*Oct 1 16:50:18.749:   AS 106, Flags 0x0, Seq 0/0 idbQ 0/0 iidbQ
un/rely 0/0
*Oct 1 16:50:18.753: %DUAL-5-NBRCHANGE: IP-EIGRP(0) 106: Neighbor
192.168.10.1 (Serial0/0) is down: Auth failure
```

Între cele două rutere nu se mai poate stabili o adiacență:

```
R2#show ip eigrp neighbors
IP-EIGRP neighbors for process 106
```

4.4 Studiu de caz

Pentru a evidenția modul de configurare EIGRP, se propune următorul scenariu. Compania AgiliCat urmează să conecteze toate sediile deținute. Pentru infrastructura de rețea, o cerință crucială este timpul de raspuns foarte mic şi convergența cât mai rapidă a protocolului de rutare în cazul apariției unor probleme.

Știind că protocolul EIGRP, în cazul unei implementări eficiente, poate răspunde instant la modificări apărute în topologie, s-a luat decizia de a-l folosi în rețeaua companiei. În implementarea soluției se va urmări în primul rând micşorarea timpului de răspuns la modificări ale rețelei, precum şi optimizarea informației din tabelele de rutare, prin manipularea rutelor. În figura 4-9 este prezentată topologia peste care va trebui să ruleze protocolul.

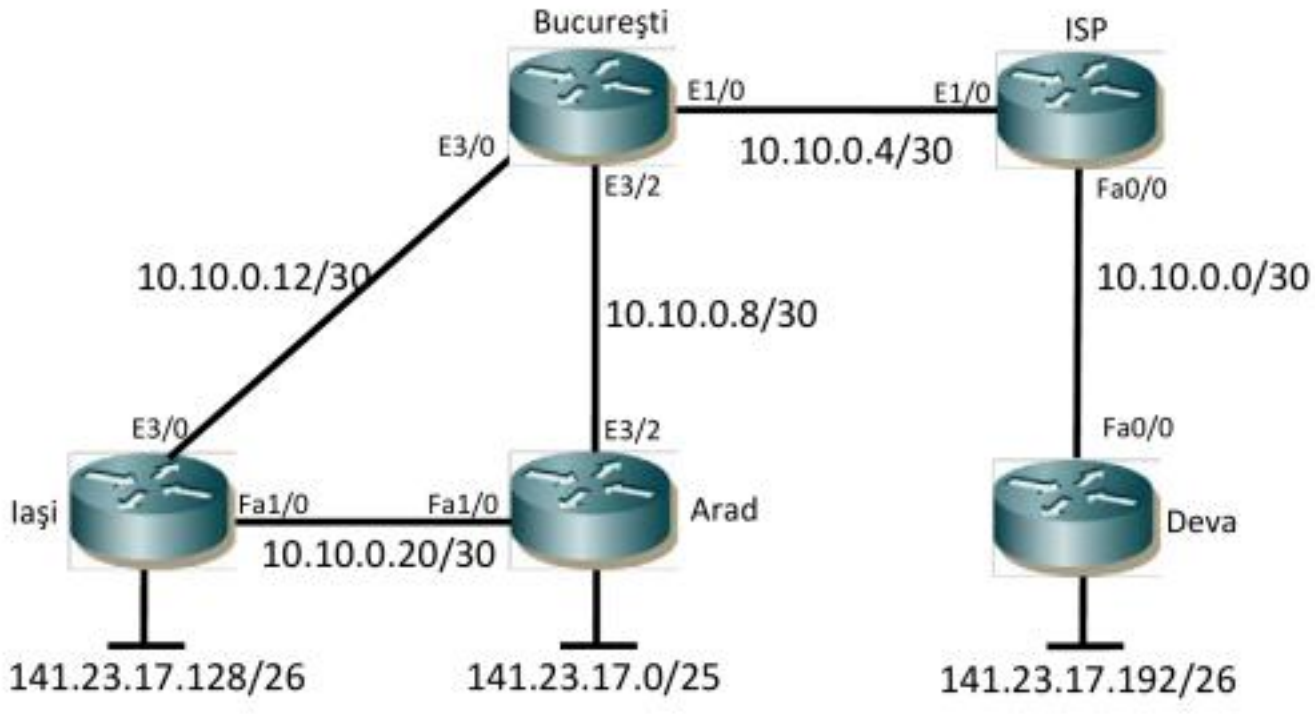

Figura 4-11: Topologia rețelei companiei AgiliCat

4.4.1 Configurări de bază

Ruterele fac parte din acelaşi sistem autonom: 24012. Compania are alocat spațiul de adrese 141.23.17.0/24, subnetat precum se vede în schema topologiei. Pentru legăturile dintre rutere se foloseşte spațiul de adrese 10.10.0.0/24. Ruterul ISP nu aparține companiei, el fiind folosit pentru redistribuirea informațiilor EIGRP în protocolul folosit în interiorul ISP şi nu prezintă interes din punct de vedere al configurării. Configurația IP a interfețelor este prezentată mai jos:

```
ISP#show ip interface brief | include up
FastEthernet0/0            10.10.0.1       YES manual up                    up
Ethernet1/0                10.10.0.5       YES manual up                    up

Bucuresti#show ip interface brief | include up
Ethernet1/0                10.10.0.6       YES manual up                    up
Ethernet3/0                10.10.0.13      YES manual up                    up
Ethernet3/2                10.10.0.9       YES manual up                    up

Arad#show ip interface brief | include up
Ethernet0/0                10.10.0.10      YES manual up                    up
FastEthernet1/0            10.10.0.21      YES manual up                    up
Loopback0                  141.23.17.1     YES manual up                    up

Iasi#show ip interface brief | include up
FastEthernet1/0            10.10.0.22      YES manual up                    up
Ethernet3/0                10.10.0.14      YES manual up                    up
Loopback0                  141.23.17.129    YES manual up                    up

Deva#show ip interface brief | include up
FastEthernet0/0            10.10.0.2       YES manual up                    up
Loopback0                  141.23.17.193   YES manual up                    up
```

Se activează procesul EIGRP 24012 pe toate ruterele. Se dezactivează sumarizarea automată pe care o face protocolul. Pe routerul ISP se generează o rută default ce va fi apoi transmisă celorlalte rutere. Se foloseste Null0 ca interfață de ieşire pentru a simula o legătură la Internet. Pachetele ce merg pe ruta default vor fi de fapt aruncate.

```
ISP(config)#ip route 0.0.0.0 0.0.0.0 null0
ISP(config)#router eigrp 24012
ISP(config-router)#network 10.10.0.0 0.0.0.3
ISP(config-router)#network 10.10.0.4 0.0.0.3
ISP(config-router)#no auto-summary
ISP(config-router)#redistribute static

Bucuresti(config)#router eigrp 24012
Bucuresti(config-router)#network 10.10.0.8 0.0.0.3
Bucuresti(config-router)#network 10.10.0.4 0.0.0.3
Bucuresti(config-router)#network 10.10.0.12 0.0.0.3
Bucuresti(config-router)#no auto-summary

Arad(config)#router eigrp 24012
Arad(config-router)#network 10.10.0.8 0.0.0.3
Arad(config-router)#network 10.10.0.20 0.0.0.3
Arad(config-router)#network 141.23.17.0 0.0.0.127
Arad(config-router)#no auto-summary

Iasi(config)#router eigrp 24012
Iasi(config-router)#network 10.10.0.20 0.0.0.3
Iasi(config-router)#network 10.10.0.12 0.0.0.3
Iasi(config-router)#network 141.23.17.128 0.0.0.63
Iasi(config-router)#no auto-summary

Deva(config)#router eigrp 24012
Deve(config-router)#network 10.10.0.0 0.0.0.3
Deva(config-router)#network 141.23.17.192 0.0.0.63
Deva(config-router)#no auto-summary
```

Pentru a verifica funcționarea corectă a protocolului de rutare, se pot verifica foarte rapid adiacențele care s-au format între vecini:

```
Iasi#show ip eigrp neighbors
IP-EIGRP neighbors for process 24012
H   Address                 Interface       Hold Uptime   SRTT   RTO  Q  Seq
                                            (sec)         (ms)       Cnt Num
1   10.10.0.13              Et3/0             14 00:07:37  404  2424  0  29
0   10.10.0.21              Fa1/0             10 00:07:50  320  1920  0  25
```

Se observă din output-ul de mai sus că routerul Iași a stabilit adiacențe EIGRP cu vecinii săi, București și Arad. Verificarea se poate continua asemănător pentru toate routerele.

După ce s-a constatat că procesul a fost pornit cu succes pe toate echipamentele, trebuie inspectată tabela de rutare pentru a vedea dacă toate rutele care ne așteptăm să existe au fost propagate.

```
Bucuresti#sh ip route
Gateway of last resort is not set

     141.23.0.0/16 is variably subnetted, 3 subnets, 2 masks
D       141.23.17.128/26 [90/409600] via 10.10.0.14, 00:00:24, Ethernet3/0
D       141.23.17.192/26 [90/412160] via 10.10.0.5, 00:00:12, Ethernet1/0
D       141.23.17.0/25 [90/409600] via 10.10.0.10, 00:01:05, Ethernet3/2
     10.0.0.0/30 is subnetted, 5 subnets
D       10.10.0.0 [90/284160] via 10.10.0.5, 00:00:12, Ethernet1/0
C       10.10.0.4 is directly connected, Ethernet1/0
C       10.10.0.8 is directly connected, Ethernet3/2
C       10.10.0.12 is directly connected, Ethernet3/0
D       10.10.0.20 [90/284160] via 10.10.0.14, 00:00:12, Ethernet3/0
                   [90/284160] via 10.10.0.10, 00:00:12, Ethernet3/2
D*EX 0.0.0.0/0 [170/281600] via 10.10.0.5, 00:00:04, Ethernet1/0
```

Rutarea fiind asigurată, este momentul să ne punem problema optimizării timpilor de convergență a rețelei.

4.4.2 Optimizarea timpilor de convergență folosind DUAL

Protocolul EIGRP permite reținerea în tabela de topologie a unor rute de rezervă, denumite rute viabile, ce vor promova în tabela de rutare automat, când nici un ruter succesor nu mai este accesibil. Aceste rute vor folosi ca următor hop rutere denumite succesori viabili (Feasible Successor - FS).

Trebuie să manipulăm metrica anumitor rute astfel încât să respecte condiția de FS. Astfel, pentru ruterul Bucureşti trebuie ca interfața Loopback de pe Iaşi să fie cunoscută direct via Ethernet 3/0 şi, ca rută de rezervă, prin ruterul Arad, via Ethernet 3/2.

Pentru ruterul Iaşi trebuie garantată ieşirea către ISP prin Arad în eventualitatea în care ruta directă prin Bucureşti nu mai este accesibilă.

Vom analiza în primul rând tabela de topologie EIGRP pentru ruterul Bucureşti, pentru a vedea care este starea rutelor ce ne interesează.

```
Bucuresti#sh ip eigrp topology
IP-EIGRP Topology Table for AS(24012)/ID(10.10.0.17)
[...]
P 141.23.17.64/26, 1 successors, FD is 409600
        via 10.10.0.14 (409600/128256), Ethernet3/0
        via 10.10.0.10 (412160/156160), Ethernet3/2
```

Se pot face o serie de observații privind rezultatul acestei comenzi. În primul rând, ruta este marcată ca P(assive), ceea ce înseamnă că este stabilă, nu se face în acest moment o recalculare a sa şi că a fost propusă spre promovare către tabela de rutare.

Se precizează apoi că fiecare rută are un succesor, asta înseamnă că un singur router este folosit în acest moment pentru trimiterea mai departe a pachetelor.

Totuşi, sunt precizate două rute către rețeaua 141.23.17.64. Una dintre ele, cea de cost total mai mic (40960) este ruta directă de la Bucureşti la Iaşi, a doua fiind ruta de rezervă prin Arad. Faptul că această rută de rezervă apare în tabela de topologie înseamnă că ea va fi folosită în cazul în care ruta mai bună nu mai este accesibilă. Se observă că este respectată regula de alegere a FS, costul total al celei mai bune rute fiind mai mare decât distanța raportată de primul vecin pe ruta de rezervă (409600 > 156160).

Se poate trage de aici concluzia că ruterul Bucureşti are deja o rută de rezervă către rețeaua 141.23.17.64/26.

Se pune acum aceeaşi problemă pentru ruta default de pe routerul Iaşi. Se va analiza tabela de topologie EIGRP pentru a vedea dacă ruta prin Arad a fost promovată ca rută de rezervă.

```
Iasi#show ip eigrp topology
IP-EIGRP Topology Table for AS(24012)/ID(141.23.17.65)
Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
       r - reply Status, s - sia Status

P 0.0.0.0/0, 1 successors, FD is 307200
        via 10.10.0.13 (307200/281600), Ethernet3/0
```

Se observă că ruta prin Arad nu îndeplineşte constrângerea de metrică pentru a putea fi promovată ca rută de rezervă. În acest caz, va trebui alterată metrica rutelor pentru a îndeplini condiția.

Cea mai simplă metodă de a face acest lucru este creşterea costului rutei directe către Bucureşti, astfel încât noua metrică a acestei rute să fie mai mare decât metrica raportată de Arad. Pentru a creşte costul rutei către Bucureşti, o metodă rapidă este declararea unei lățimi de bandă mai mică pe interfața dintre Iaşi şi Bucureşti, ceea ce va aduce o penalizare de cost rutei directe către Bucureşti.

```
Iasi(config)#interface ethernet 3/0
Iasi(config-if)#bandwidth 9900
```

În acest moment, a fost declarată o lățime de bandă de doar 9.9Mb/s pe legătura de 10Mb/s Eth3/0. Doar lățimea de bandă declarată a fost modificată, traficul pe interfață nefiind limitat în nici un fel la 9Mb/s.

Trebuie văzut acum rezultatul în tabela de topologie pe care l-a avut configurația făcută.

```
Iasi#show ip eigrp topology
Iasi#sh ip eigrp topology
IP-EIGRP Topology Table for AS(24012)/ID(141.23.17.65)
Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
       r - reply Status, s - sia Status

P 0.0.0.0/0, 2 successors, FD is 309760
        via 10.10.0.13 (309760/281600), Ethernet3/0
        via 10.10.0.21 (309760/307200), FastEthernet1/0
```

Se observă că noua metrică a legăturii directe către Bucureşti este egală cu metrica prin Arad. În acest moment, routerul Iaşi are doi succesori către 0.0.0.0/0 şi va putea face balansare pe cele două căi. Dacă nu ne dorim acest comportament, se poate creşte la 9901kb/s lățimea de bandă pe

interfață pentru ca ruta folosită să fie cea prin Bucureşti şi cea prin Arad să fie de rezervă, sau se poate declara o lățime de bandă de 8999kb/s pentru ca ruta prin Arad să fie cea principală.

4.4.3 Optimizarea timpilor de convergență folosind rețele Stub

În cazul în care o rețea nu mai este accesibilă şi nu există rută de rezervă, procesul EIGRP plasează ruta respectivă în starea Active şi caută o altă cale de a ajunge la rețea. Această căutare se face trimițând vecinilor săi pachete prin care aceştia sunt interogați cu privire la o eventuală rută alternativă către rețeaua în cauză.

Totuşi, pentru o anumită categorie de rutere, răspunsul la aceste interogări va fi tot timpul că nu dispun de o cale alternativă. Acestea sunt routerele Stub, exemplu fiind routerul Deva din topologie. Aceste rutere sunt identificabile prin faptul că au un singur vecin cu care schimbă informații de rutare, în cazul de față informații EIGRP.

Pentru a nu se mai consuma timp şi resurse interogând acest router, el poate fi declarat ca Stub, ca în exemplul următor:

```
Deva(config)#router eigrp 24012
Deva(config-router)#eigrp stub connected summary
```

Opțiunile de "connected" şi "summary" permit routerului Stub să trimită vecinilor săi informații despre rețelele direct conectate şi să sumarizeze rețele (fie manual, fie automat).

Pentru a vedea rezultatul comenzii date anterior, pe routerul local se foloseşte comanda:

```
Deva#show ip protocols
Routing Protocol is "eigrp 24012"
[...]
  EIGRP stub, connected, summary
```

Pe vecinul routerului Deva, ISP, acest lucru este vizibil inspectând vecinii EIGRP:

```
ISP#show ip eigrp neighbors detail
IP-EIGRP neighbors for process 24012
H   Address                 Interface       Hold Uptime   SRTT   RTO  Q  Seq
                                            (sec)         (ms)       Cnt Num
0   10.10.0.2               Fa1/0             10 00:10:10  128   768  0  90
   Version 12.4/1.2, Retrans: 0, Retries: 0, Prefixes: 2
   Stub Peer Advertising ( CONNECTED SUMMARY ) Routes
   Suppressing queries
```

Se observă că ISP are activată opțiunea de "suppressing queries" (nu se trimit interogări) către ruterul Deva.

4.4.4 Minimizarea dimensiunii tabelei de rutare

Următoarea discuție pleacă de la tabela de rutare a routerului ISP.

```
ISP#show ip route
Gateway of last resort is 0.0.0.0 to network 0.0.0.0

  141.23.0.0/16 is variably subnetted, 3 subnets, 2 masks
D     141.23.17.128/26 [90/435200] via 10.10.0.6, 00:31:35, Ethernet1/0
D     141.23.17.192/26 [90/156160] via 10.10.0.2, 00:10:12, FastEthernet0/0
D     141.23.17.0/25 [90/435200] via 10.10.0.6, 01:51:21, Ethernet1/0
  10.0.0.0/30 is subnetted, 5 subnets
C     10.10.0.0 is directly connected, FastEthernet0/0
C     10.10.0.4 is directly connected, Ethernet1/0
D     10.10.0.8 [90/307200] via 10.10.0.6, 03:35:03, Ethernet1/0
D     10.10.0.12 [90/307200] via 10.10.0.6, 03:07:54, Ethernet1/0
D     10.10.0.20 [90/309760] via 10.10.0.6, 03:37:11, Ethernet1/0
S* 0.0.0.0/0 is directly connected, Null0
```

Dacă rețelele din 10.0.0.0/8 fac parte dintr-un spațiu privat de adrese şi vor fi filtrate pentru a nu ajunge in Internet, nu la fel stau lucrurile cu rețelele din 141.23.0.0/16. Rutele către aceste rețele urmând să fie trimise mai departe către vecini, trebuie avut grijă ca ele să fie sumarizate cât mai atent, pentru a micşora dimensiunea tabelei de rutare.

Sumarizarea poate introduce însă probleme la rândul ei, dacă ruterul ce face sumarizarea nu are rute către toate părțile componente ale supranetului.

De exemplu, pentru topologia de față, rutele către 141.23.17.0/25 şi 141.23.17.128/26 pot fi sumarizate pe ruterul Bucureşti la 141.23.17.0/24. Acest lucru se realizează ca în exemplul următor:

```
Bucuresti(config)#interface ethernet 1/0
Bucuresti(config-if)#ip summary-address eigrp 24012 141.23.17.0
255.255.255.0
```

Comanda de sumarizare este dată în modul interfață, rețeaua sumarizată fiind trimisă doar vecinului/vecinilor de pe interfața respectivă. În acest moment, tabela de rutare a routerului Bucureşti s-a modificat astfel:

```
Bucuresti#show ip route

Gateway of last resort is 10.10.0.5 to network 0.0.0.0

     141.23.0.0/16 is variably subnetted, 3 subnets, 3 masks
D       141.23.17.128/26 [90/409600] via 10.10.0.14, 00:57:52, Ethernet3/0
D       141.23.17.192/26 [90/412160] via 10.10.0.5, 00:59:12, Ethernet1/0
D       141.23.17.0/25 [90/409600] via 10.10.0.10, 00:59:12, Ethernet3/2
D       141.23.17.0/24 is a summary, 00:01:27, Null0
     10.0.0.0/30 is subnetted, 5 subnets
D       10.10.0.0 [90/284160] via 10.10.0.5, 00:59:12, Ethernet1/0
C       10.10.0.4 is directly connected, Ethernet1/0
C       10.10.0.8 is directly connected, Ethernet3/2
C       10.10.0.12 is directly connected, Ethernet3/0
D       10.10.0.20 [90/284160] via 10.10.0.14, 00:57:54, Ethernet3/0
                   [90/284160] via 10.10.0.10, 00:57:54, Ethernet3/2
D*EX 0.0.0.0/0 [170/281600] via 10.10.0.5, 00:59:12, Ethernet1/0
```

În tabela de rutare a apărut o rută către 141.23.17.0/24 ce are ca următor hop Null0, adică pachetele vor fi aruncate.

Totuşi, acest lucru nu se întâmplă datorită faptului că tabela de rutare este sortată descrescător după dimensiunea măştii. Astfel, se observă că cele 3 rute reale către rețelele din 141.23.17.0/24 vor fi verificate înainte de ruta sumarizată şi pachetele vor fi rutate corect de către Bucureşti.

Rutele către toate rețelele din tabela de rutare care sunt incluse în supra-rețea sunt înlocuite cu ruta care duce către Null0. Astfel, Bucureşti nu va mai trimite către ISP rute către 141.23.17.0/25 sau 141.23.17.128/26, ci doar către 141.23.17.0/24. Tabela de rutare a lui ISP va arăta astfel:

```
ISP#show ip route

Gateway of last resort is 0.0.0.0 to network 0.0.0.0

   141.23.0.0/16 is variably subnetted, 3 subnets, 3 masks
D     141.23.17.192/26 [90/156160] via 10.10.0.2, 00:10:12, FastEthernet0/0
D     141.23.17.0/24 [90/435200] via 10.10.0.6, 00:17:44, Ethernet1/0
   10.0.0.0/30 is subnetted, 5 subnets
C     10.10.0.0 is directly connected, FastEthernet0/0
C     10.10.0.4 is directly connected, Ethernet1/0
D     10.10.0.8 [90/307200] via 10.10.0.6, 04:17:40, Ethernet1/0
D     10.10.0.12 [90/307200] via 10.10.0.6, 03:50:31, Ethernet1/0
D     10.10.0.20 [90/309760] via 10.10.0.6, 04:19:48, Ethernet1/0
S* 0.0.0.0/0 is directly connected, Null0
```

5 OSPF

5.1 Prezentarea protocolului

5.1.1 Link state vs. distance vector

Prima soluție de rutare dinamică standardizată de IETF a fost protocolul RIP. Chiar şi prin optimizările aduse de versiunea a doua, limitările de scalabilitate au determinat căutarea unei soluții adecvate rețelelor de mari dimensiuni. O soluție eficientă în termeni de consum de bandă şi timp de convergență trebuia să se îndepărteze de la paradigma protocoalelor bazate pe vectori de distanță.

OSPF este un protocol bazat pe starea conexiunilor (*link-state*), standardizat de IETF la începutul anilor `90. Acesta este un protocol open standard: OSPF v.1 a fost definit în 1995 prin RFC 1850, modificat apoi în OSPF v.2 prin RFC 2328 (1998), iar în anul 2008 RFC 5340 a actualizat protocolul la versiunea 3, introducând schimbări semnificative pentru adaptarea la IPv6.

Caracteristicile importante ce recomandă OSPF în fața RIP țin de diferențele tradiționale dintre protocoalele *distance vector* şi cele *link-state*. Astfel, deşi RIP poate oferi soluții eficiente de rutare pentru rețele de mici dimensiuni, creşterea complexității rețelei se traduce în penalități însemnate asupra timpului de convergență. OSPF oferă un timp de convergență de ordinul secundelor, cel mai adesea cu două ordine de mărime mai mic decât cel oferit de RIP.

Funcționalitatea RIP în rețelele mari şi eterogene este afectată de limitarea capacității sale de rutare la căi de maximum 15 hopuri, orice destinație mai îndepărtată fiind considerată inaccesibilă. În schimb protocolul OSPF nu prezintă această limitare.

De asemenea, RIP v.1 nu este compatibil cu Variable Length Subnet Masks (VLSM), spre deosebire de OSPF, care permite adresarea VLSM şi CIDR (Classless Inter-Domain Routing). OSPF va trimite în actualizări şi masca de rețea, pe lângă rețeaua destinație.

În cazul OSPF, după stabilirea adiacenței cu vecinii şi schimbarea informațiilor de rutare, actualizările pentru protocoalele link-state se fac incremental, precizând doar schimbările apărute, spre deosebire de protocoalele *distance vector* ce vor trimite informații complete de rutare vecinilor la fiecare actualizare. Transmiterea periodică a tabelei de rutare în întreaga rețea conduce la o convergență lentă. În plus, într-o rețea stabilă, consumul de bandă pentru schimbarea informațiilor

de rutare este mult mai scăzut pentru OSPF, actualizările fiind trimise doar în cazul unor schimbări de topologie, şi nu la intervale fixe de timp, precum în cazul RIP.

Protocoalele *distance vector* sunt vulnerabile în cazul modificărilor din rețea, deoarece în timpul în care se realizează convergența creşte probabilitatea unor rutări suboptimale sau a unor bucle infinite. Intervalul cel mai vulnerabil pentru OSPF este perioada inițială de descoperire a rețelei, în care actualizările LSA pot ocupa o proporție considerabilă a lățimii de bandă. În ansamblu, însă, OSPF este mult mai stabil în fața schimbărilor de topologie, datorită faptului că fiecare ruter are o imagine de ansamblu a rețelei şi rutează pachetele independent, conform propriilor decizii.

Principalul avantaj al protocoalelor *distance vector* îl reprezintă complexitatea redusă a configurării şi întreținerii unei astfel de soluții. În plus, cerințele hardware în termeni de memorie şi timp de procesare sunt semnificativ mai reduse pentru un protocol de tip *distance vector*.

O altă limitare a protocoalelor bazate pe vectori distanță constă în faptul că ruterele nu au o imagine de ansamblu asupra topologiei rețelei, deoarece procesează doar informațiile conținute în tabelele de rutare ale vecinilor direct conectați. Ruterele care rulează OSPF obțin o imagine completă asupra rețelei, denumită în continuare tabela de topologie, dat fiind că acumulează informații de topologie de la toate celelalte rutere.

Deoarece ruterul care rulează OSPF dispune de o reprezentare completă a topologiei rețelei, va selecta cea mai bună rută conform unui algoritm Dijkstra Shortest Path First (SPF). Rutele sunt ierarhizate conform costului (metricii) acestora, fiind selectată ruta cu costul cel mai redus. Implicit, costul unei interfețe este calculat ca fiind invers proporțional cu lățimea de bandă a acesteia; în acest caz, este folosită valoarea declarată a lățimii de bandă, nu valoarea reală. Totuşi, este posibil ca metrica rețelei să fie setată explicit la o valoare anume, ceea ce permite optimizarea deciziilor ruterului în funcție de situația reală a rețelei. Desigur, este posibil ca ruta cu cel mai mic număr de hopuri, care ar fi fost selectată de un protocol *distance vector*, să aibă un cost mai mare decât o rută mai lungă dar care traversează legături cu lățime de bandă mai mare.

Un alt avantaj al OSPF față de RIP este prevenirea buclelor de trafic. Acestea pot apărea atunci când modificările din topologia rețelei se propagă lent. De exemplu, un ruter neinformat de eşecul unui segment din rețea poate trimite pachetele pe o cale ce nu mai este valabilă, în timp ce ruterul următor de pe această cale le va trimite înapoi, fiind informat despre schimbare, pentru a ajunge la destinație pe o altă cale. Bucla de trafic poate dispărea fie dacă primul ruter ajunge la o reprezentare adecvată a căilor spre destinație, fie dacă sunt implementate mecanisme adiționale pentru a preveni, detecta şi elimina buclele de trafic. În cazul OSPF, propagarea rapidă a informațiilor privind modificările din topologia rețelei previne apariția unor astfel de situații, astfel încât rutele calculate de OSPF nu conțin bucle de trafic.

Rețelele RIP sunt plate deoarece nu există conceptul de zone, în timp ce OSPF suporta atât implementări plate: single-area (SA), cât şi implementări ierarhice: multi-area (MA) permitând împărțirea logică în zone pentru a limita inundarea întregii rețele cu actualizări atunci când se modifică topologia.

OSPF funcționează diferit în funcție de tipul de rețea, definit ca fiind unul dintre următoarele trei: rețele punct-la-punct, rețele multiacces broadcast, precum Ethernet, şi rețele multiacces fără capacitate de broadcast, precum Frame Relay sau ATM. Toate rețelele multiacces au ca trăsătură comună necesitatea alegerii unui ruter responsabil cu actualizările (un DR şi un BDR, conform procesului descris în continuare).

Performanțele OSPF în ceea ce priveşte convergența şi latența sunt datorate caracteristicilor cheie precum independența deciziilor ruterelor şi organizarea ierarhică a rețelei. În continuare va fi

analizată funcționarea OSPF şi rolul acestor trăsături, pornind de la etapa inițială a formării relațiilor de adiacență, până la selectarea rutelor optime şi actualizarea informațiilor de rutare.

5.1.2 Funcționarea protocolului

Funcționarea procesului OSPF pe un ruter se desfăşoară în cinci etape:

1. identificarea vecinilor ruterului şi stabilirea relațiilor de adiacență;
2. în cazul rețelelor multiacces: selectarea unui ruter responsabil de actualizări şi a unui ruter de backup;
3. descoperirea rutelor;
4. selectarea rutelor optime către destinație;
5. actualizarea informațiilor de rutare.

Înainte de a face schimb de informații, două rutere vecine trebuie să se identifice şi să stabilească o relație de adiacență. Acest proces are loc cu ajutorul pachetelor de tip *Hello*, prin care se stabilesc şi se mențin relații bidirecționale. O relație bidirecțională cu un anumit vecin este stabilită în momentul în care un ruter se recunoaşte pe sine în lista vecinilor din pachetul *Hello* primit de la acel vecin.

Un pachet *Hello* conține următoarele informații: identificatorul ruterului, intervalele Hello şi Dead, lista vecinilor, identificatorul zonei, prioritatea ruterului, adrese IP ale ruterelor DR si BDR, informația de autentificare şi marcajul de zonă stub.

Fiecare ruter are un **identificator OSPF**, constând dintr-un număr de 32 de biți, exprimat în general în forma zecimală specifică adreselor IP. Alegerea OSPF_ID se va face la inițializarea procesului OSPF. Acest identificator poate fi configurat explicit. Dacă nu există o astfel de setare, identificatorul OSPF va fi cea mai mare adresă IP a interfețelor de loopback; dacă nu există nici interfețe de loopback configurate, valoarea OSPF_ID va fi inițializată cu valoarea celei mai mari adrese IP dintre adresele interfețelor active. Acest identificator este folosit atât la stabilirea relațiilor bidirecționale, cât şi în schimbul de mesaje de actualizare.

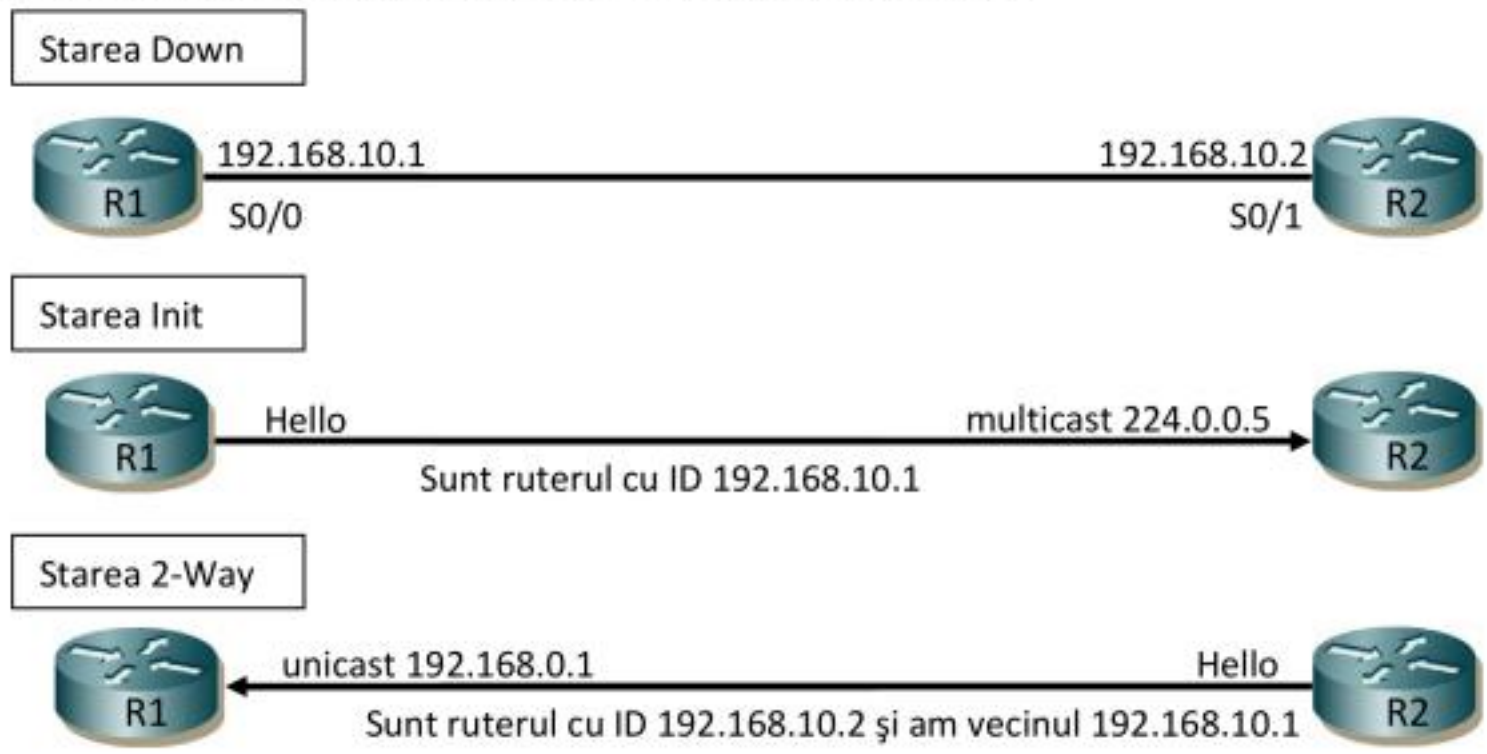

Figura 5-1: Stabilirea comunicației bidirecționale

Odată cu pornirea unui proces OSPF pe un ruter, acesta trece din starea **Down** în starea **Init**, trimițând pachete Hello către vecini. Spre deosebire de protocoalele *distance vector* care folosesc pachete de broadcast, OSPF inițiază procesul de identificare a vecinilor folosind adresa multicast specifică 224.0.0.5. Configurația implicită în rețelele Ethernet presupune trimiterea de mesaje *Hello*

către ruterele vecine la fiecare 10 secunde (intervalul *Hello*). Ruterele vecine ce rulează OSPF vor adăuga ruterul care a inițiat comunicația în tabela de vecini şi îi vor răspunde cu un pachet *Hello*, de tip unicast. După cum se observă în figura 5-1, adresa IP a ruterului care a inițiat procesul de adiacență este inclusă de altfel în pachetul Hello de răspuns, pentru a confirma validitatea schimbului de mesaje. În momentul recepționării răspunsului ruterul trece în starea **2-Way**. În această stare, ruterul completează o tabelă de adiacență cu informații privind toți vecinii cu care au stabilit o legătură bidirecțională.

În rețelele multiacces precum Ethernet, Frame-Relay sau ATM, urmează etapa de selectare a **ruterului desemnat** pentru actualizări (DR – Designated Router) şi a ruterului de backup (Backup Designated Router – BDR). Cele două rutere au rolul de a media schimbul informațiilor de actualizare. Mai precis, toate actualizările sunt trimise doar către DR şi BDR, nu şi către restul ruterelor din rețeaua multiacces. Dacă informația din pachetul de actualizare este nouă, DR va propaga mesajul către toate ruterele. Rolul BDR este de a păstra adiacențe cu toate ruterele din rețea (ca şi DR), şi de a prelua rolul de DR dacă acesta din urmă devine inactiv.

DR şi BDR sunt selectate pe baza priorității OSPF atribuite ruterului. Prioritatea unui ruter de a deveni DR se stabileşte la nivel de interfață, putând fi configurată între 0 şi 255. Prioritatea implicită este 1, iar valorile mai mari sunt preferate în selecție. Prioritatea 0 indică faptul că ruterul nu este eligibil pentru a deveni DR sau BDR. Dacă ruterele au aceeaşi prioritate, de exemplu valoarea 1, va fi ales ca DR ruterul cu cel mai mare OSPF_ID. Ruterul BDR este cel cu următoarea valoare în ordine descrescătoare. Alegerea DR este nepreemptivă: dacă un ruter cu o prioritate mai mare sau cu o valoare mai mare a OSPF_ID devine activ după ce un alt DR a fost ales deja, alegerea rămâne stabilă până în momentul în care DR devine inactiv. În această situație BDR promovează şi se reface alegerea pentru un nou ruter de backup. Este de asemenea important de observat că statutul de DR şi BDR se referă de fapt la interfața unui ruter, nu la ruterul propriu-zis (un ruter poate fi DR pe un segment de retea, BDR pe un altul, si poate sa nu aiba nici unul dintre aceste roluri pe alte segmente multiacces direct conectate).

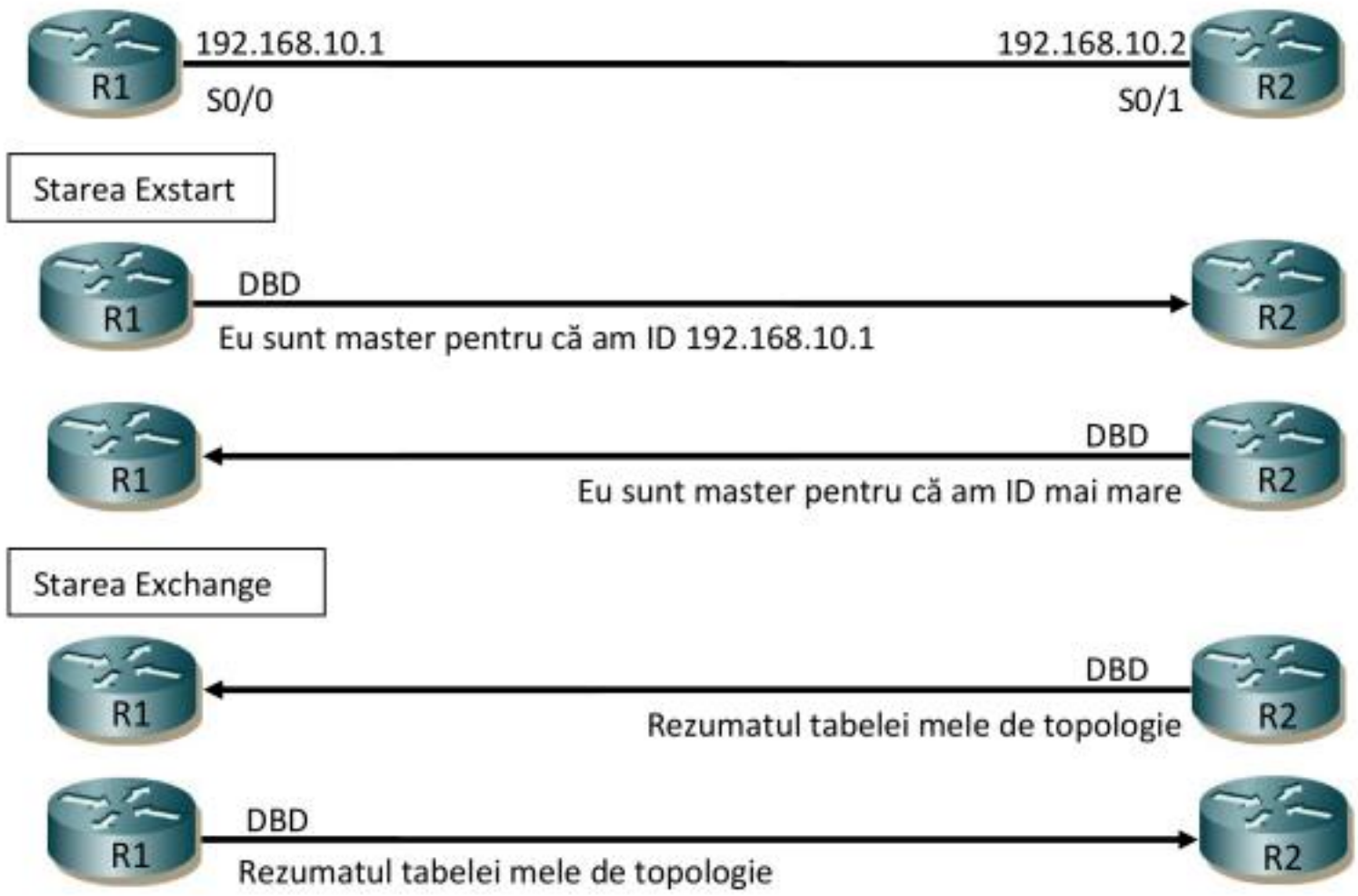

Figura 5-2: Descoperirea rutelor

După ce a fost finalizată alegerea ruterului desemnat, procesul OSPF intră în starea **Ex-start** (Exchange Start) şi iniţiază relaţia de comunicare cu DR şi BDR, relaţia având o structură asimetrică de *master-slave*. Această ierarhie este necesară pentru a stabili iniţiatorul comunicării (master), cel care va incrementa numărul de secvenţă al pachetelor de actualizare. Ruterul *master* este cel cu cea mai mare adresă IP; acesta poate fi ruterul DR dar şi un alt ruter, deoarece este posibil ca DR să fi fost selectat pe baza priorităţii, de exemplu, şi să aibă o adresă IP mai mică, ajungând în ipostaza de *slave* în relaţia cu unul dintre vecini.

După cum se observă în figura 5-2, odată stabilite rolurile în comunicare, ruterele adiacente schimbă pachete Database Description (DBD) în starea **Exchange**, trimiţând astfel informaţii condensate privind starea tabelei de topologie a fiecăruia. Ruterul master va incrementa numărul de secvenţă a pachetelor DBD pe măsură ce sunt trimise, permiţând detectarea schimbărilor recente în informaţiile privind topologia reţelei. Dacă un ruter observă, pe baza informaţiilor din pachetele DBD, că tabela sa conţine date ce nu mai sunt de actualitate, solicită primirea pachetelor de update (LSU), ca în figura 5-3. Ruterul intră apoi în starea **Loading**, de schimb de informaţii privind starea legăturilor din reţea. Odată ce tabelele de topologie sunt sincronizate, ruterele ajung în starea **Full** de adiacenţă deplină. În cazul reţelelor Ethernet, starea Full se realizează doar în relaţia dintre un ruter şi vecinii DR şi BDR respectivi. Toate celelalte relaţii de vecinătate vor fi conservate în starea 2-Way.

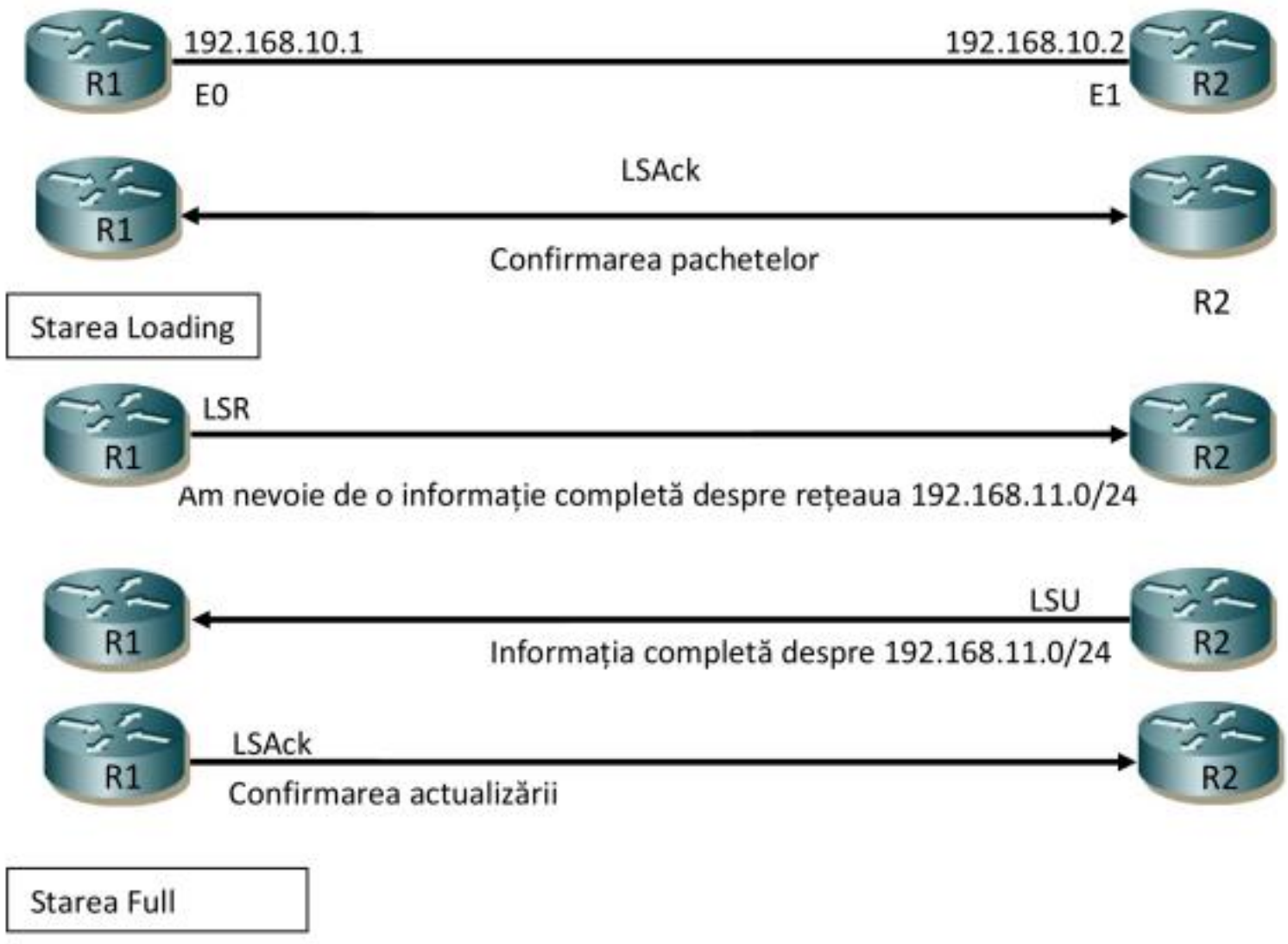

Figura 5-3: Descoperirea rutelor

Stare	Acțiuni caracteristice
Down	Ruterul nu a trimis pachete Hello pentru o perioadă mai lungă decât intervalul permis (denumit interval *Dead*), sau ruterul a fost eliminat manual din configurația rețelei. Ruterele pot trimite pachete *Hello* către vecinii aflați în starea *Down*.
Init	Ruterul a inițiat o comunicație prin trimiterea unui mesaj *Hello* pe adresa multicast
2-Way	Ruterul a stabilit o relație bidirecțională cu vecinul său, ca urmare a primirii unui răspuns Hello valid (identificat prin prezența adresei sale IP) sau a unui pachet *Database Descriptor* (DBD). Ruterul decide dacă rămâne în această stare sau dacă își modifică relația de vecinătate la nivelul Full.
Ex-start	Stabilirea statutului de master / slave pentru inițierea comunicării între rutere; în cazul rețelelor multiacces statutul se stabileşte în relația dintre rutere şi DR, respectiv BDR.
Exchange	În principal are loc schimbul de pachete DBD.
Loading	Pe baza informațiilor din DBD sunt solicitate actualizări ale stării legăturilor, fiind transmise pachete LSR, LSU şi LSAck.
Full	Stare de adiacență deplină: informațiile de rutare sunt sincronizate.

Figura 5-4: Starile protocolului OSPF

5.1.3 Tabelele OSPF

Informațiile de care dispune procesul OSPF sunt structurate în trei tabele, şi anume tabela de adiacență, tabela de topologie şi tabela de rutare.

Ca urmare a stabilirii relațiilor bidirecționale cu ruterele vecine ce rulează OSPF, un ruter va completa **tabela de adiacență** (*Adjacency database* sau *Neighbor table*). Dacă unul dintre vecini nu trimite pachete *Hello* pentru o perioadă mai lungă decât intervalul permis (interval Dead) tabela de adiacență este actualizată prin ştergerea acestuia şi se invalidează toate căile care treceau prin acel vecin. Intervalul *Dead* este setat implicit cu o valoare de patru ori mai mare decât intervalul la care se trimit pachetele *Hello*.

Tabela de topologie sau tabela topologică (*Link-state database* – LSDB, sau *Topological database*) conține o imagine detaliată a întregii rețele, constând în toate mesajele de actualizare a stării legăturilor, mai exact stările tuturor legăturilor ruterelor din rețea. LSDB este sincronizată periodic cu tabelele celorlalte rutere din rețea, sau din aria respectivă – în cazul OSPF Multi-area, ceea ce înseamnă ca ruterele au o tabelă de topologie identică. Fiecare proces OSPF dispune de o tabelă de topologie, pe baza căreia ia deciziile de selectare a căilor optime ce vor fi promovate în tabela de rutare.

În cazul OSPF Single Area, pentru a selecta ruta optimă către o destinație, ruterul aplică algoritmul Dijkstra SPF, situându-se la rădăcina arborelui şi căutând ruta cu metrica cea mai scăzută. Ruta optimă este adăugată la **tabela de rutare** (*Routing table* sau *Forwarding database*). În cazul primirii unei actualizări de la un ruter vecin, tabela de topologie este modificată iar algoritmul SPF este rulat pentru recalcularea rutelor optime. Fiecare ruter are o tabelă de rutare unică, ce reflectă poziția sa în topologie.

În cazul în care rețeaua este segmentată în arii diferite, fiind implementat OSPF Multi-area, construirea tabelei de rutare se realizează diferențiat pentru destinațiile din propria zonă şi cele din zonele externe. În primul rând, ruterul va calcula rute optime către destinațiile din zona în care se află, şi le va adăuga în tabela de rutare. Apoi, ruterul va prelucra informațiile privind rețele aflate în alte zone OSPF şi va selecta căile de cost minim. În final sunt analizate destinațiile externe rețelei OSPF. De asemenea, în cazul în care un ruter dispune de o ruta inter-zonală şi o rută intra-zonală către o aceeaşi destinație, va fi păstrată ruta intra-zonală. În cel de-al treilea pas, ruterul va calcula

rutele optime către destinațiile aflate în afara rețelei OSPF, cu excepția situației în care este situat într-o arie de tipul stub sau totally stubby (definite mai jos).

5.1.4 Formatul pachetelor OSPF

În subcapitolul 5.1.2 au fost prezentate etapele funcționării protocolului OSPF. Conform datelor prezentate anterior, există 5 tipuri de pachete OSPF, folosite pentru stabilirea relațiilor de adiacență şi pentru actualizarea tabelei topologice. În timp ce adiacența este realizată prin mesaje Hello, care au un TTL egal cu 1, actualizarea se realizează prin comunicarea de informații Link-State Advertisement (LSA), incluse în pachete Link-State Update (LSU). Solicitarea unei LSA se realizează prin pachete LSR – Link State Request, iar confirmarea primirii se realizează prin pachete Link-State Acknowledgement (LSAck). Pachetele DBD (Database Description) reprezintă o prezentare succintă a tabelei topologice a unui ruter, incluzând lista tuturor ruterelor cunoscute şi ultimul număr de secvență pentru fiecare.

Pachetele OSPF sunt încapsulate direct în câmpul de date IP, fără a apela la un alt protocol de nivel transport precum TCP sau UDP; prin urmare, OSPF va folosi propria schemă de confirmare a recepționării mesajelor, bazată pe un pachet special, şi anume LSAck. Identificatorul protocolului OSPF în antetul IP este 89.

Câmp		Informații
Versiunea		Versiunea 2 sau 3
Tipul		Identifică unul dintre cele 5 tipuri de pachete
Lungime		Lungimea pachetului în octeți
ID-ul ruterului		Identifică ruterul sursă al pachetului
ID-ul ariei		Identifică aria sursă a pachetului
Sumă de control		Valoare folosită pentru a verifica integritatea pachetului
Tipul autentificării		Specifică una dintre cele trei posibilități: nici o autentificare, parole în clar, sau mesaje criptate Message Digest 5 (MD5)
Autentificarea		Folosit în procesul autentificării
Date	Hello	Lista vecinilor cunoscuți
	DBD	Rezumatul tabelei de topologie
	LSR	Tipul de LSU solicitat şi ID-ul ruterului care dispune de informație
	LSU	Informația completă de actualizare, inclusiv pentru mai multe LSA
	LSAck	Câmp gol

Figura 5-5: Structura unui pachet OSPF

Actualizările sunt trimise fie ca urmare a sesizării unor modificări în topologia rețelei, fie ca urmare a expirării informațiilor. Mesajele de actualizare LSA conțin un câmp de vârstă, ceea ce permite expirarea lor după o anumită durată prestabilită, setată implicit la 1800 de secunde. Ruterul care a emis actualizarea expirată o retrimite, cu un număr de secvență incrementat, într-un pachet LSU, ce poate include mai multe intrări LSA.

De asemenea, ruterul care observă o modificare a topologiei rețelei informează ruterele vecine, sau, în cazul rețelelor multiacces, informează DR care ulterior trimite mesajul către ruterele din aria respectivă (LSA flood). Fiecare ruter compară LSA primit cu cele deja disponibile în tabela de topologie; dacă informația este nouă, o adaugă; dacă un LSA similar este deja înregistrat, compară numărul de secvență şi păstrează înregistrarea cea mai recentă.

OSPF foloseşte adresarea multicast şi unicast. De exemplu, odată ce un ruter a sesizat modificarea topologiei, va trimite pachete LSU către adresa de multicast 224.0.0.5 (pentru All Link State Routers) – adresă folosită şi în starea Init de stabilire a relațiilor de adiacență prin transmiterea pachetelor *Hello*.

În cazul rețelelor multiacces, ruterul care a sesizat o modificare va trimite o actualizare pe adresa de multicast 224.0.0.6, adresă pe care ascultă doar DR şi BDR. În cazul în care pachetul de actualizare reprezintă o informație nouă, DR va retrimite pachetul tuturor vecinilor pe adresa 224.0.0.5.

5.1.5 OSPF Multi-area

Scalabilitatea OSPF se datorează în mare măsură posibilității de a segmenta rețeaua în mai multe arii, cu avantaje semnificative pentru timpul de convergență al protocolului şi timpul de răspuns al rutării.

Cu cât o rețea este mai mare, cu atât tabela topologică şi tabela de rutare conțin mai multe intrări. Numărul de căi potențiale spre aceeaşi destinație creşte odată cu dimensiunea rețelei, în consecință calculele SPF necesare obținerii rutei optime devin mai complexe. Astfel creşte consumul de resurse de memorie şi procesor dar şi de lățime de bandă din cauza actualizărilor mai frecvente. In plus, datorita modului de functionare al protocoalelor link-state, orice schimbare in retea, va fi propagata catre toate ruterele care trebuie sa recalculeze fiecare legatura din intreaga tabela de topologie.

Pentru a simplifica reprezentările topologice şi calculul rutelor optime, rețeaua poate fi structurată în zone (sau arii). **O arie** reprezintă o mulțime logică de rutere, legături şi rețele OSPF, fiind reprezentată printr-un identificator propriu. În cazul OSPF Multi-Area, ruterele vor menține o tabelă topologică detaliată doar pentru aria în care sunt situate, ceea ce reduce substanțial dimensiunea LSDB precum şi cantitatea de informație propagată prin LSA. Mai precis, propagarea LSA detaliate poate fi restrânsă la granițele ariei, ceea ce înseamnă că schimbările topologice vor afecta doar zona în care apar. Ruterele vor trebui să cunoască doar informațiile despre vecinii aflați în aceeaşi arie, informațiile despre restul rețelei putând fi sumarizate. Nu în ultimul rând, segmentarea pe arii permite reducerea dimensiunea tabelei de rutare prin sumarizare, precum şi a frecvenței rulării algoritmului SPF.

Identificatorul unei arii este un număr exprimat pe 32 de biți, cel mai adesea sub forma unei singure valori în notație zecimală. O practică frecventă este ca ariile să fie identificate în funcție de adresa IP a principalului ruter. Totuşi, identificatorul unei arii nu este o adresă IP, din punct de vedere funcțional, şi prin urmare poate coincide cu o adresă IP fără a crea confuzii.

OSPF Multi-area se bazează pe un model ierarhic pe doua nivele. Aria 0 (supranumita si aria de backbone sau aria de tranzit) este intotdeauna zona centrala; toate celelalte arii se ataseaza la aceasta. Aria 0 formeaza nivelul de sus al ierarhiei, iar celelalte zone ramase, nivelul de jos; acest design ofera suport pentru sumarizare – intern posibila doar intre arii si pentru minimizarea numarul din tabela de rutare. **Zonele normale** sunt cele care includ utilizatorii finali, fiind definite în special după criterii utilitariste, de exemplu pe criterii geografice sau organizaționale. De regula, o zonă normală nu este folosita pentru a transporta trafic intre zone. Un exemplu de topologie împărțită pe zone este reprezentată în figura 5-6.

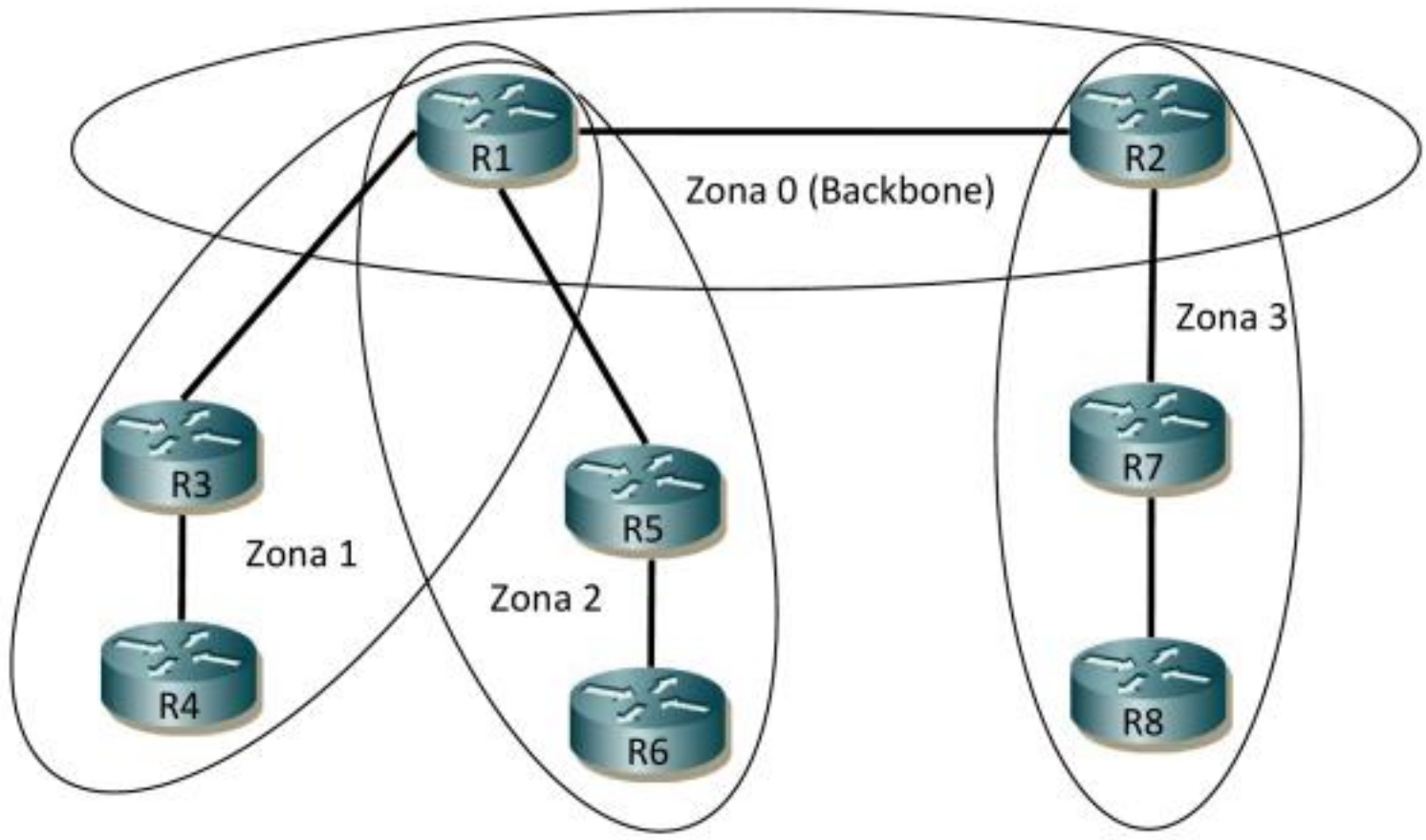

Figura 5-6: Organizarea ierarhică

Ruterele OSPF Multi-area pot fi clasificate în funcție de rolul lor în comunicarea intra- şi inter-arii. Un **ruter intern** (*Internal Router*) este cel care, după cum sugerează şi numele, are toate interfețele în aceeaşi zonă. Un ruter de margine, abreviat **ABR** (*Area Border Router*) are interfețe în două sau mai multe zone, pe care astfel le conectează. Un **ASBR** (*Autonomous System Boundary Router*) conectează un alt protocol de rutare la rețeaua OSPF. Denumirea poate fi înşelatoare, deoarece OSPF nu este folosit pentru a ruta între sisteme autonome diferite, fiind un IGP. Prin urmare, orice ruter care distribuie un alt protocol de rutare într-o rețea OSPF este un ASBR. **Ruterele de backbone** (*Backbone Router*) au cel puțin o interfață în aria 0, participând astfel la conectarea zonelor normale cu zona 0. Un ruter poate îndeplini roluri multiple: de exemplu, un ruter backbone poate fi un Internal Router, un ABR sau un ASBR. Ruterele care conectează mai multe zone păstrează câte o tabelă de topologie distinctă pentru fiecare dintre ele. Pe de altă parte, toate ruterele dintr-o arie trebuie să aibă o tabelă de topologie identică.

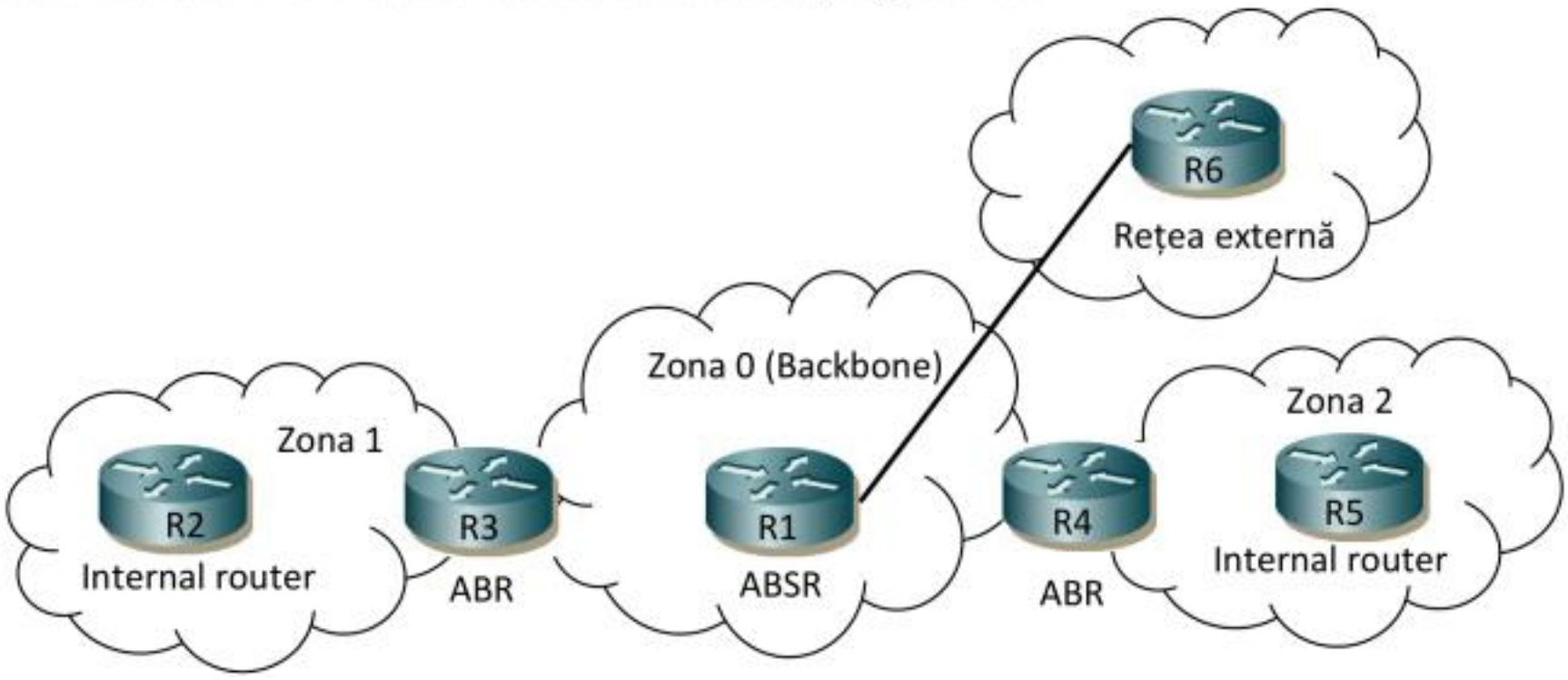

Figura 5-7: Tipuri de rutere OSPF

Proiectarea unei rețele OSPF ierarhice trebuie să țină cont de resursele disponibile la nivelul fiecărei zone definite (precum memorie sau CPU), precum şi de tipul legăturilor, stabilitatea acestora şi a dispozitivelor de interconectare, şi de alte particularități ale rețelei. Există nişte recomandări generale ale Cisco privind configurarea OSPF Multi-area, precum: un număr maxim de 50 de rutere în fiecare zonă, un număr maxim de 60 de vecini pentru fiecare ruter, cel mult 3 zone per ruter, şi evitarea situațiilor în care un ruter este ales DR sau BDR în două sau mai multe LAN-uri. Desigur, aceste recomandări trebuie avute în vedere în contextul mai larg definit de resursele disponibile, de cerințele utilizatorilor şi de designul de ansamblu al rețelei.

5.1.6 Tipuri de LSA

Comunicarea în cazul OSPF Multi-area se realizează prin mai multe tipuri de mesaje LSA, clasificate în funcție de ruterul care le inițiază şi de informația pe care o conțin. Mesajele LSA primite de un ruter formează tabela sa de topologie (LSDB). Ele descriu întreaga topologie a unei rețele sau a unei zone OSPF.

LSA de tipul 1, **Router Link LSA**, sunt generate de fiecare ruter, sunt transmise doar în interiorul ariei, şi informează toate ruterele din aria respectivă cu privire la starea legăturilor ruterului sursă. Pentru fiecare legătură sunt precizate tipul, ID-ul ruterului următor, adresa interfeței ruterului, precum şi metrica. LSA de tip 1 nu trec dincolo de ABR.

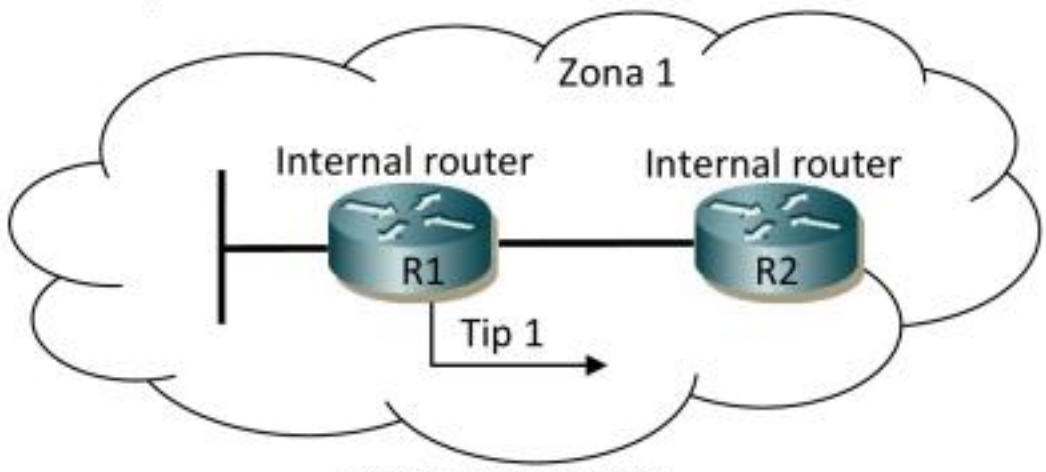

Figura 5-8: LSA tip 1

LSA de tipul 2, **Network Link LSA,** este generat pentru o rețea de tip multiacces, de către DR. LSA2 conține o listă a identificatorilor OSPF a tuturor ruterelor legate la respectiva rețea multiacces. Altfel spus, un LSA2 conține toate adresele ruterele din rețea cu care DR are stabilită o relație de adiacență, inclusiv acesta. Ca şi LSA1, LSA2 nu trece de granițele ariei.

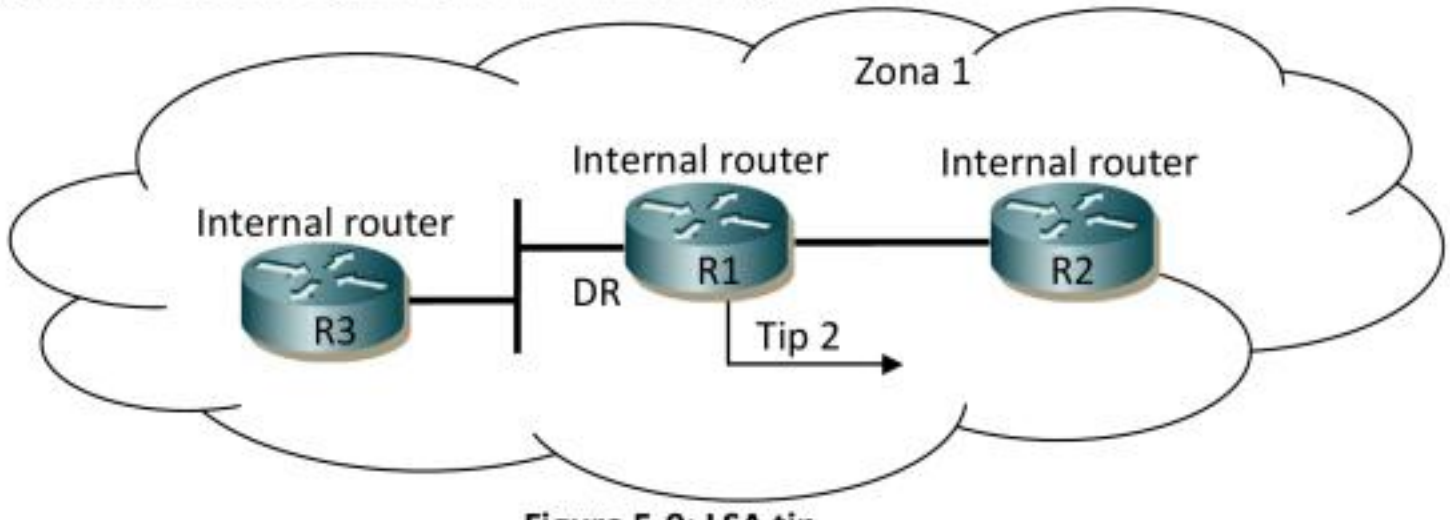

Figura 5-9: LSA tip

LSA de tipul 3, ***Network Summary* LSA**, este inițiat de ABR şi conține informațiile propagate anterior prin intermediul LSA1 şi LSA2 cu privire la rețelele incluse în aria respectivă. Mai precis, LSA3 descrie rețelele de destinație conectate cu ABR, specificând masca acestora precum şi metrica.

Informațiile sunt trimise în zona 0 către celelalte ABR, pentru a propaga rutele către zonele locale conectate la ABR-ul inițiator. Ruterele de graniță vor regenera LSA3 şi vor transmite apoi informațiile în interiorul zonelor din care fac parte. Astfel, informațiile dintr-un LSA3 sunt propagate în întregul sistem autonom OSPF. Setarea implicită a ABR este ca informația din LSA3 să nu fie sumarizată, ceea ce poate duce la o creştere semnificativă a traficului de actualizare. LSA3 pot include însă şi informații sumarizate manual, ceea ce este recomandabil pentru a spori scalabilitatea.

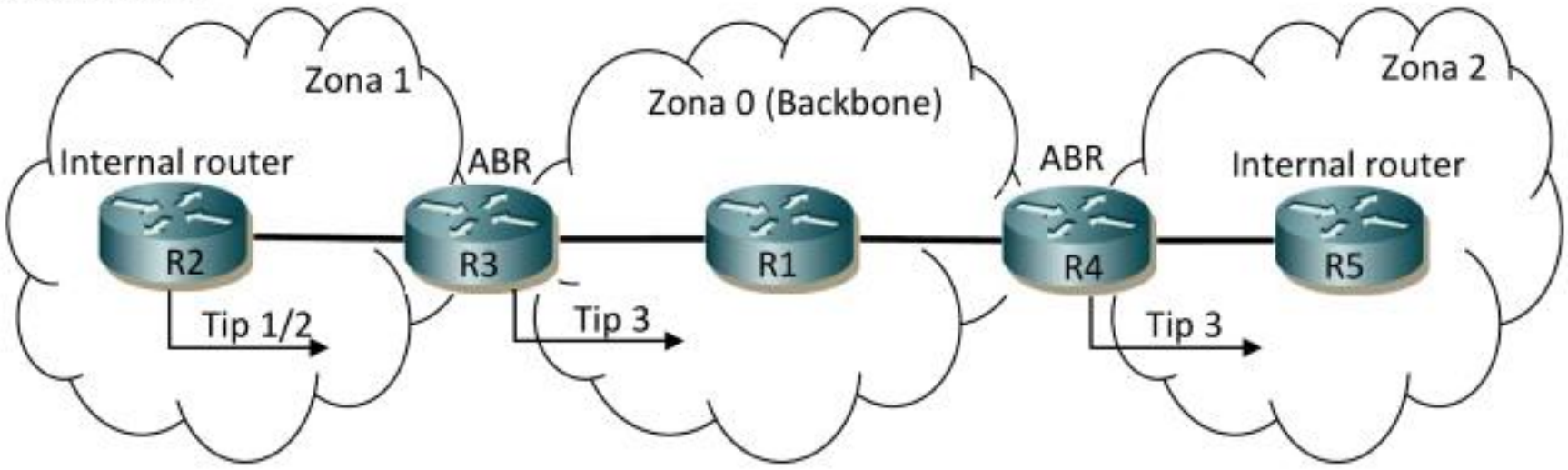

Figura 5-10: LSA tip 3

LSA de tipul 4, **Network Summary LSA**, este generat de un ruter ABR la primirea unui LSA1 ce descrie un ASBR. LSA de tipul 4 identifică ruterul ASBR şi oferă o rută către el. Pentru tot traficul rutat în exteriorul sistemului autonom, sunt necesare informații despre ruterul ASBR care oferă accesul către rutele externe. Ruterul ASBR din figura 5-11 trimite un LSA1 cu bitul extern setat prin care se identifică a fi ASBR. Când un ABR primeşte un astfel de LSA, construieşte un LSA4 şi îl trimite în zona 0 (Backbone). Următoarele rutere ABR vor regenera LSA4 şi îl vor trimite în zonele la care sunt conectate. Aceste LSA vor conține identificatorul ruterului ASBR care a trimis primul LSA1.

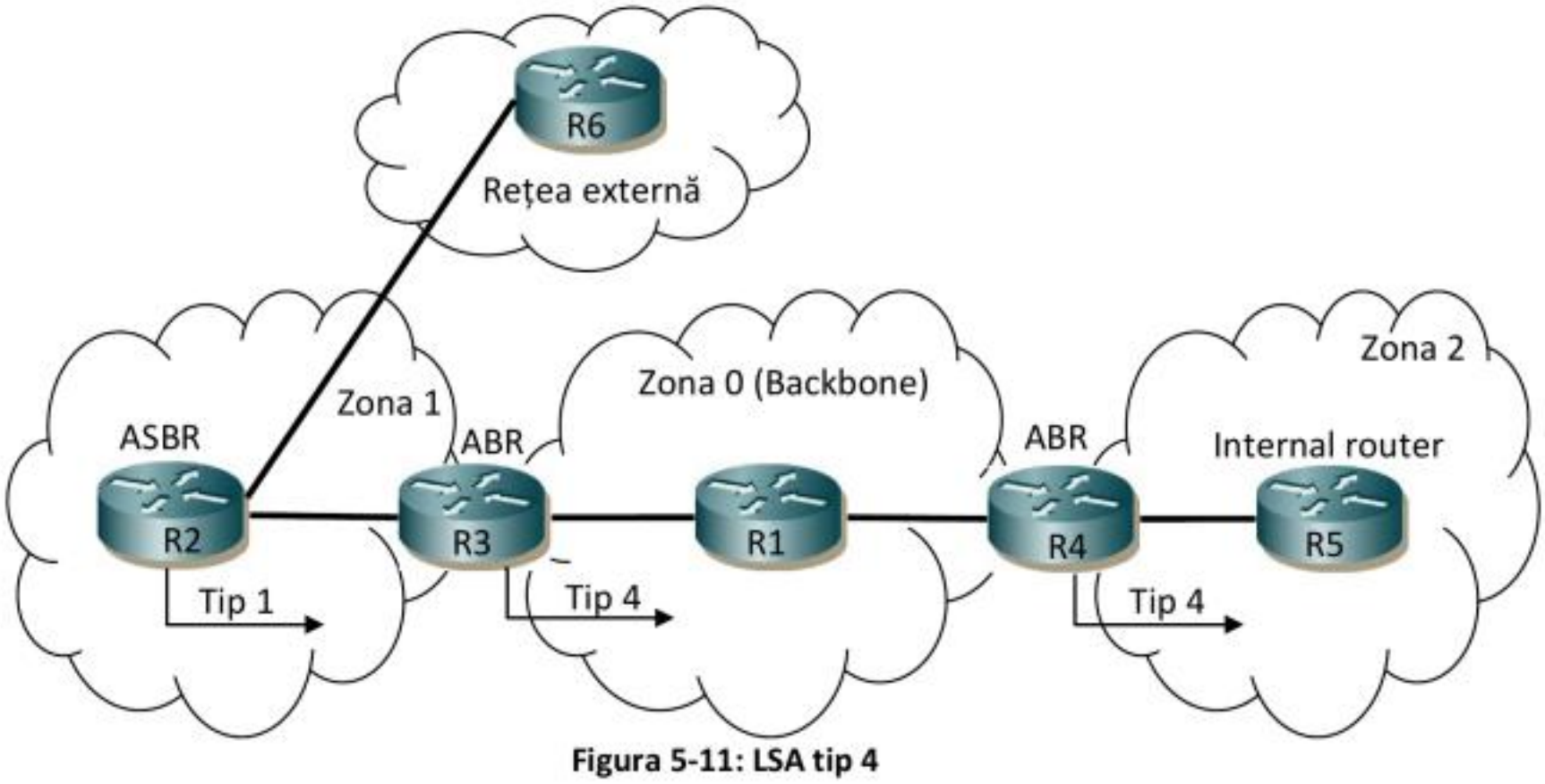

Figura 5-11: LSA tip 4

LSA de tipul 5, **External Link LSA**, descrie o rută către o rețea externă sistemului autonom OSPF. LSA5 este generat de către ASBR şi este propagat în tot sistemul autonom. Identificatorul este cel al rețelei externe şi nu se schimbă pe parcursul propagării prin rețea. Sumarizarea pentru rutele externe nu este activată în mod implicit, dar se recomandă sumarizarea lor la nivelul ruterului ASBR.

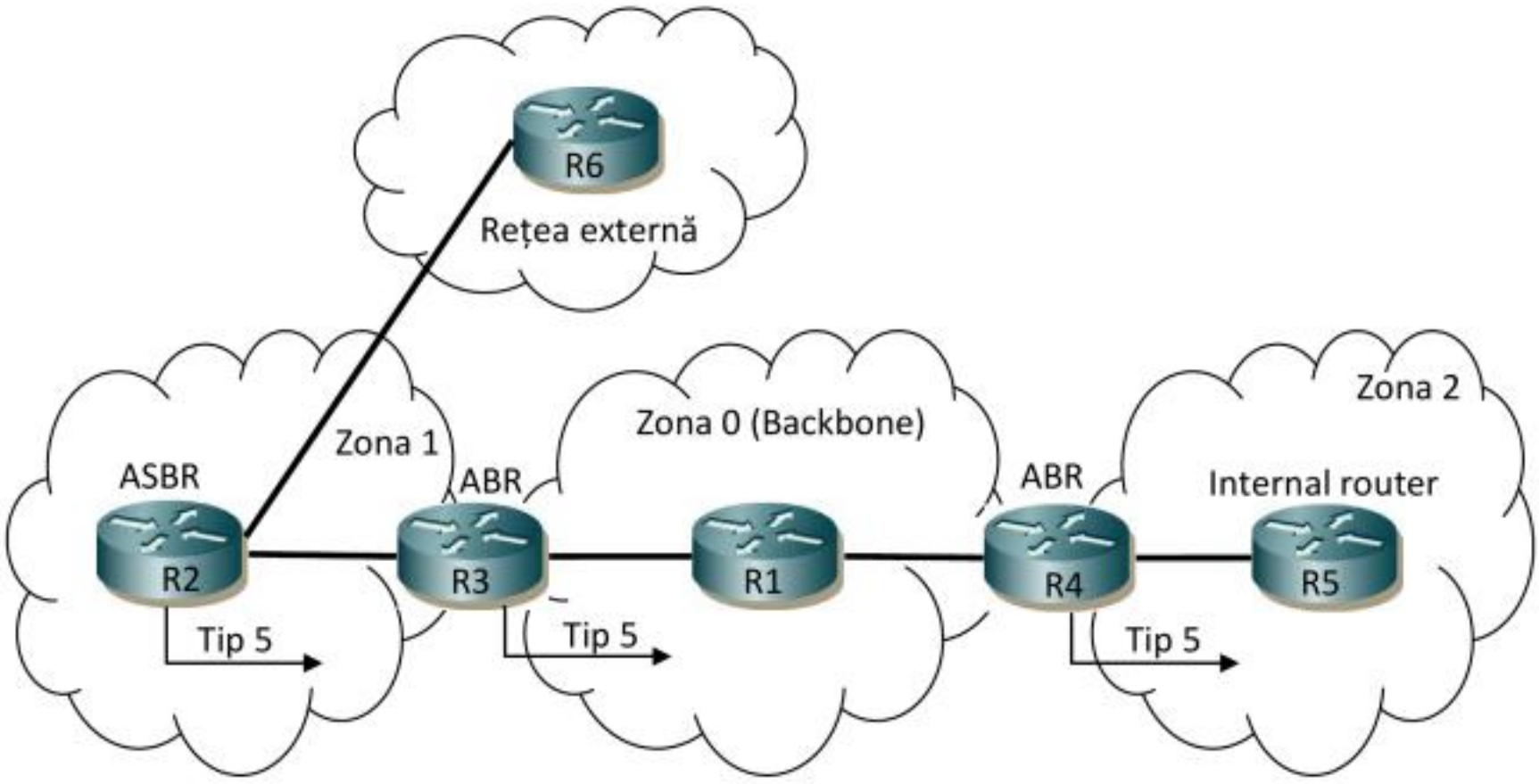

Figura 5-12: LSA tip 5

Costul unei rute externe depinde de tipul configurat pe ASBR. Există două tipuri de rute externe în ceea ce privește costul: tipul O E1 și tipul O E2. Pentru tipul O E1, costul se calculează adăugându-se la costul extern, costul fiecărei legături pe care pachetul o traversează. Acest tip se folosește atunci când mai multe rutere ASBR anunță rute către aceeași destinație, pentru a evita rutarea suboptimală. În mod implicit, este activat tipul O E2, al cărui cost este egal cu costul extern și rămâne neschimbat. Se va folosi acest tip atunci când un singur ASBR anunță o rută către o anumită destinație.

LSA de tip 6, Multicast LSA, este folosit pentru aplicații multicast, dar nu este implementat pe ruterele Cisco.

LSA de tip 7, NSSA External LSA, este generat pe ASBR în cadrul unei zone NSSA și va fi trimis doar în acea zonă. LSA7 va fi convertit în LSA5 de către ABR.

5.2 Concepte avansate

5.2.1 Tipuri de zone

OSPF facilitează scalabilitatea folosind o organizare ierarhică a comunicării. Am văzut ca zonele sunt diferențiate în zone normale și zone de tranzit, care includ și zona 0 (de backbone). La rândul lor, zonele normale sunt diferențiate în funcție de gradul și de tipul de deschidere către exterior. Există patru tipuri de zone normale: zone standard, zone Stub, zone de tip *Totally Stubby* și zone de tip *Not-So-Stubby*.

Zonele standard sunt cele care acceptă toate tipurile de update-uri și sumarizări, fără restricții. În cazul unor rețele extinse, este însă posibil ca overhead-ul introdus de LSA3, LSA4 și LSA5 să fie semnificativ.

Extinzând principiul izolării, conform căruia a fost proiectată și segmentarea rețelei OSPF în zone, unele dintre ariile normale pot fi definite ca fiind Stub sau Totally Stubby, adică arii care nu mai participă la transferul de informație detaliată privind zonele externe.

Zonele Stub nu acceptă informații cu privire la rutele din afara sistemului OSPF, și prin urmare mesajele LSA4 și LSA5 sunt filtrate. Ruterele din astfel de zone vor folosi ruta default pentru a accesa rețelele din afara sistemului OSPF. Ruterul ABR de la granița zonei Stub va genera un LSA3 pentru a anunța o rută implicită în interiorul zonei. Pentru destinațiile externe sistemului autonom se va folosi această rută default. Zonele stub nu pot conține rutere ASBR.

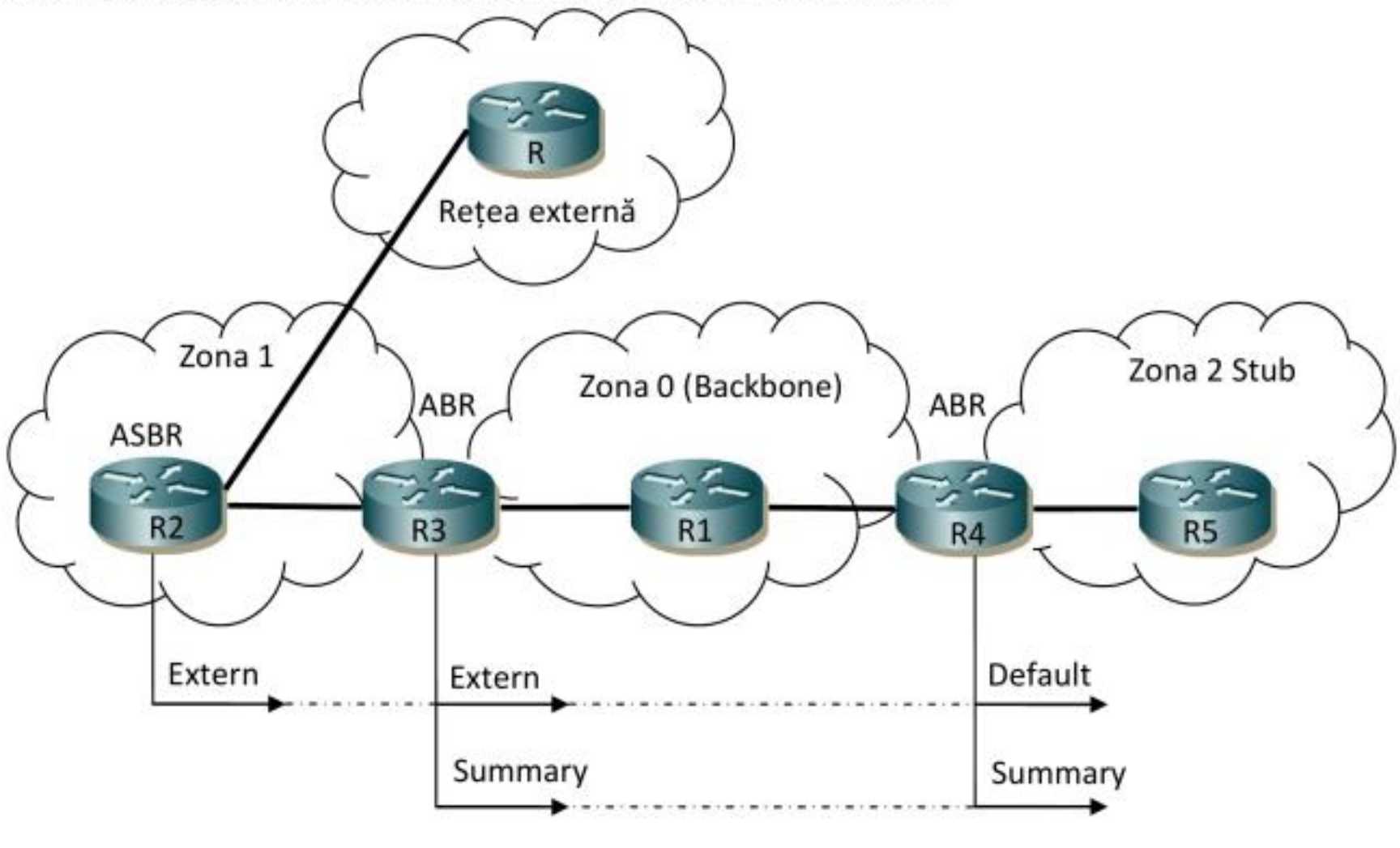

Figura 5-13: Zona Stub

La un nivel și mai ridicat de izolare, **zonele de tip Totally Stubby** nu acceptă informații cu privire la rutele externe sistemului autonom sau rute sumarizate din alte zone. În concluzie, zonele de tip *Totally Stubby* nu acceptă nici LSA3, nici LSA4 și LSA5. În ariile T*otally Stubby* sunt permise doar LSA3 care indică ruta default pentru accesul către exterior. Ruta default va fi folosită pentru a accesa toate destinațiile externe sistemului autonom sau externe acestei zone. De asemenea, zonele de tip *Totally Stubby* nu pot conține ASBR.

Zonele de tip Not So Stubby (NSSA – Not So Stubby Areas) sunt o modificare proprietară Cisco a ariilor Stub sau Totally Stubby, pentru a permite accesul acestora către rețele din afara sistemului OSPF. Prin urmare, un ASBR poate fi inclus într-o NSSA. Dat fiind că mesajele LSA5 sunt blocate în cazul ariilor Totally Stubby, NSSA importă rute externe și le comunică în interiorul său prin LSA7, care sunt apoi convertite de ABR în LSA5 și apoi propagate mai departe în rețeaua OSPF.

Zonele NSSA pot fi de două tipuri: Stub sau Totally Stubby. Comportamentul celor două tipuri de zone este identic cu cel al zonelor normale Stub și Totally Stubby, exceptând potențiala prezență a unui ASBR în zonă.

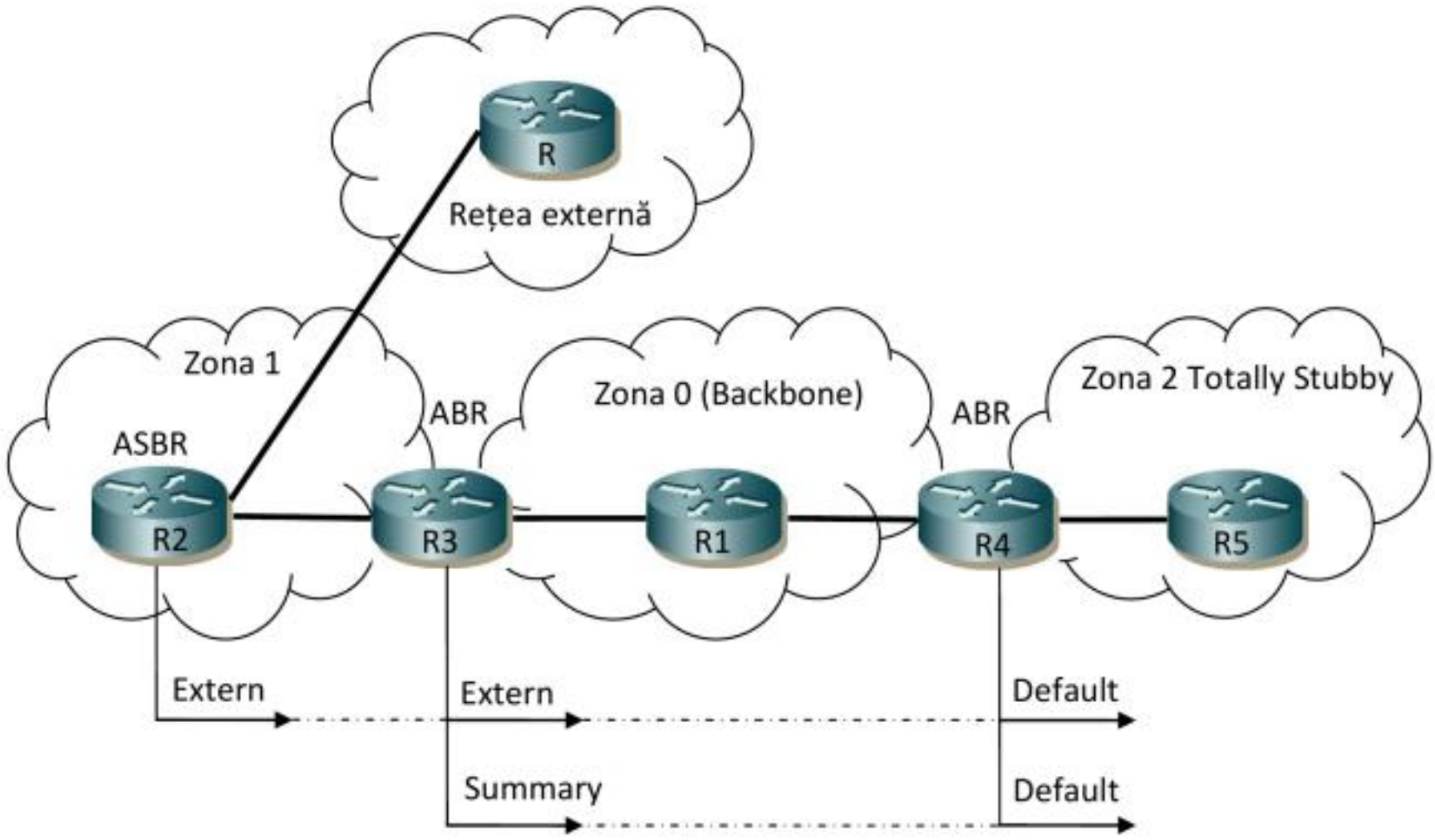

Figura 5-14: Zona Totally Stubby

Pachetele LSA de tip 7 sunt descrise în tabela de rutare ca O N1 şi O N2. N1 înseamnă că metrica este calculată ca în cazul tipului E1, iar pentru N2 metrica este calculată ca în cazul E2, iar N2 este folosit în mod implicit.

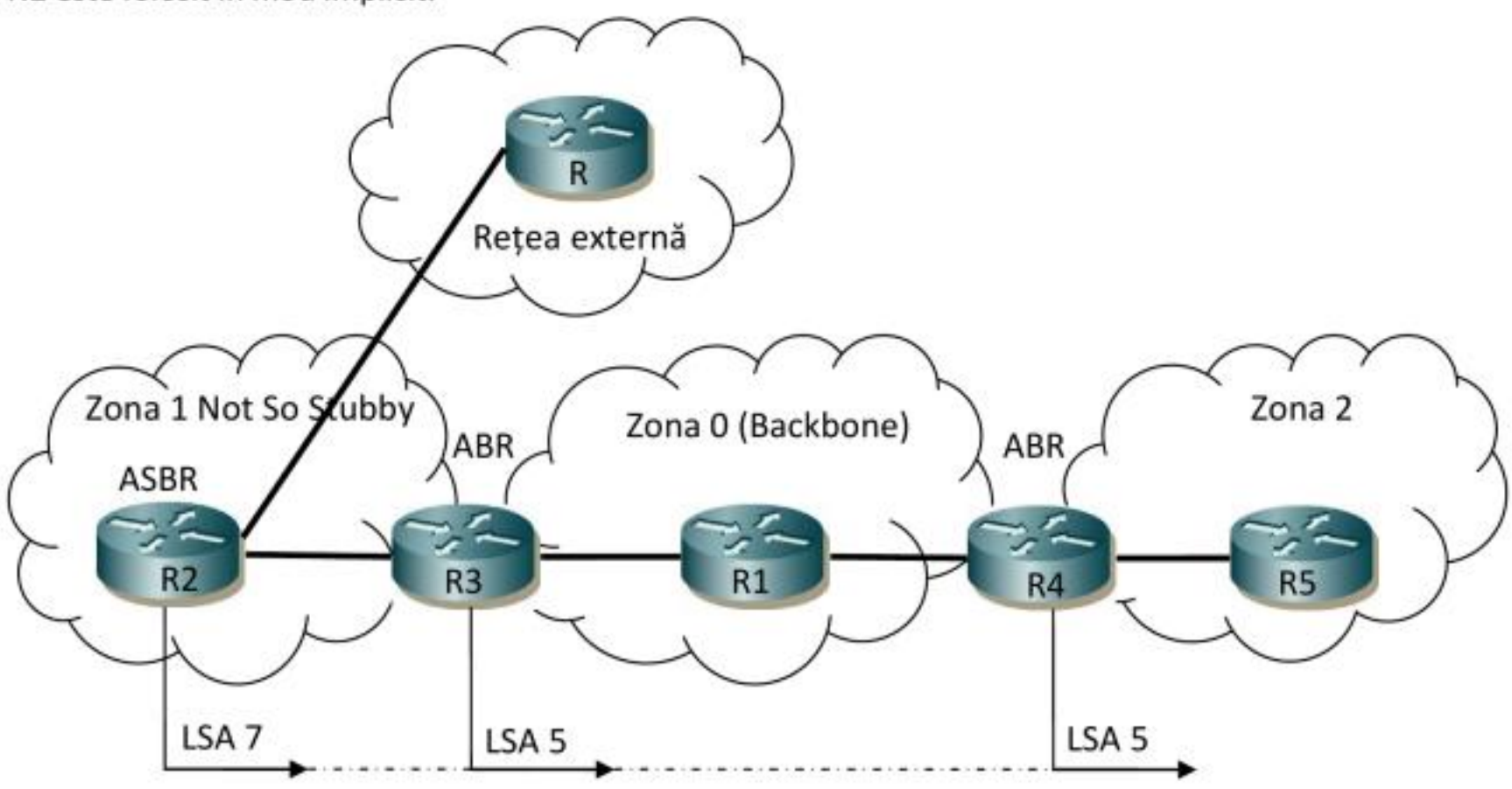

Figura 5-15: Zona Not So Stubby

Prin urmare, ariile *Stub* şi *Totally Stubby* au următoarele caracteristici distinctive: nu sunt arii de *backbone*, nu includ ASBR, şi ruterele din interiorul lor, inclusiv ABR, trebuie să fie configurate ca rutere stub înainte de stabilirea adiacențelor. De asemenea, în general ariile *Stub* şi *Totally Stubby*

sunt conectate cu restul rețelei prin intermediul unui punct de acces (ABR) unic, deşi este posibil să fie conectate prin mai multe ABR-uri, dacă rutele suboptimale sunt acceptabile.

Configurarea unei zone ca *Stub* sau *Totally Stubby* duce la micşorarea tabelei de topologie şi de rutare, ceea ce duce la consum mai mic de memorie pe ruterele din acea zonă.

De asemenea, este important ca ariile *Stub* şi *Totally Stubby* să nu fie utilizate ca arii de tranzit pentru legăturile virtuale – inclusiv pentru cele proiectate temporar sau ca soluții de backup.

5.2.2 Sumarizarea rutelor

Principalele beneficii ale sumarizării rutelor sunt creşterea scalabilității şi reducerea consumului de memorie şi procesor. O rețea OSPF poate scala până la dimensiuni destul de mari datorită sumarizării rutelor.

Sumarizarea rutelor constă în agregarea mai multor rute într-un singur anunț. Astfel se reduce numărul de pachete LSA care propagă prin rețea, dar şi dimensiunea tabelei de topologie şi de rutare, ceea ce rezultă în reducerea consumului de memorie şi procesor pe ruter.

Dacă nu se sumarizează rutele, fiecare LSA specific este propagat în zona de backbone şi apoi mai departe în celelalte zone. Atunci când ar fi generat un LSA, toate ruterele ar trebui să-şi refacă tabela de topologie şi să construiască arborele SPF.

Atunci când se sumarizează rutele, doar rute sumarizate vor fi propagate în zona de backbone. Prin urmare, atunci când o legătură devine invalidă, schimbarea de topologie nu este propagată în zona de backbone sau în celelalte zone.

OSPF pune la dispoziție două tipuri de sumarizare: între zone şi externă. **Sumarizarea rutelor între zone** diferite ale aceluiaşi sistem autonom poate fi configurată pe ABR şi se aplică pentru toate rutele din fiecare zonă. În general nu se aplică rutelor externe. Pentru sumarizarea eficientă a rutelor, rețelele trebuie să aibă asignate adrese continue.

Sumarizarea rutelor externe poate fi configurată pe ASBR şi se aplică numai rutelor externe injectate în OSPF prin redistribuție. De asemenea, este necesar ca rutele sumarizate să corespundă unor adrese de rețea continue.

5.2.3 Autentificarea în OSPF

Pentru a preveni primirea de actualizări neautorizate, este necesară folosirea autentificării vecinilor. Autentificarea se face la primirea fiecărui pachet de actualizare primit prin trimiterea unei parole cunoscute de ambele rutere care comunică. Ruterul verifică fiecare pachet OSPF primit pentru a autentifica sursa actualizării.

În mod implicit, nu este activată autentificarea în cadrul protocolului OSPF, dar acesta suportă două metode de autentificare ce pot fi configurate: o parolă simplă trimisă în clar, şi autentificarea prin MD5.

Metoda autentificării prin MD5 include folosirea unui număr de secvență în fiecare pachet OSPF pentru a proteja ruterele împotriva atacurilor de tip replay.

5.2.4 Legături virtuale

În cazul OSPF Multi-area este necesar ca toate zonele să fie conectate cu aria de backbone. În situațiile în care nu există legături fizice care să permită acest lucru, pot fi folosite legături virtuale. De exemplu, legăturile virtuale pot conecta o arie cu aria 0 prin intermediul unei zone de tranzit.În figura 5-16, zona 2 nu poate fi direct conectată la zona de backbone, deci o legătură virtuală este folosită pentru a asigura o cale către zona 0, fiind folosită zona 1 ca zonă de tranzit.

Figura 5-16: Legături virtuale

De asemenea, dacă două rețele OSPF sunt combinate într-o rețea comună, cele două arii de backbone vor putea fi unificate printr-o legătură virtuală pentru a deveni o singură arie 0. Un exemplu este reprezentat în figura 5-17.

Figura 5-17: Legături virtuale

Pachetele de tip Hello sunt trimise peste legăturile virtuale în acelaşi mod ca peste legăturile standard, la intervale de 10 secunde. Actualizările folosind pachete LSA funcţionează diferit. În general un LSA se retransmite la intervale de 30 de minute. Pachetele LSA transmise pe o legătură virtuală au setată opțiunea DoNotAge (DNA) care le împiedică să expire. Acest mecanism este folosit pentru a preveni inundarea excesivă a legăturii virtuale.

Recomandările Cisco sunt ca legăturile virtuale să nu fie planificate ca o cale stabilă de acces la aria 0, ci să fie folosite temporar, de exemplu pentru soluții de back-up, sau pentru alte situații care necesită o intervenție rapidă.

5.3 Configurarea protocolului

5.3.1 Configurări de bază

Pentru a configura protocolul de rutare OSPF este necesară parcurgerea următorilor paşi:

1. Activarea procesului OSPF pe ruter folosind comanda **`router ospf`** specificând un număr de proces unic pe ruter, spre deosebire de EIGRP pentru care trebuia specificat numarul sistemului autonom. Identificatorul procesului trebuie să fie un număr între 1 şi 65535. Numarul de proces are doar relevanță locală, de aceea nu este necesară configurarea aceluiaşi număr de proces pe toate ruterele din rețea. Rularea mai multor procese OSPF pe un singur ruter nu este recomandată din cauza consumului excesiv de memorie şi procesor.

```
R(config)# router ospf id_proces
```

2. Identificarea interfețelor care sunt incluse în procesul OSPF folosind comanda **network**. Această comandă indentifică şi zona din care face parte ruterul.

```
R(config-router)# network adresa_ip masca_wildcard area id_zona
```

Adresa rețelei este folosită pentru a determina pe ce legături să trimită actualizări, pe ce legături să asculte actualizări şi ce rețele să anunțe. Masca specificată trebuie să fie de tip wildcard. Identificatorul zonei poate fi în format decimal sau în formatul asemănător adresei IP: A.B.C.D.

O alternativă la metoda anterioară de configurare este comanda **ip ospf area** care se va introduce direct pentru o anumită interfață.

```
R(config-if)# ip ospf id_proces area id_zona [secondaries none]
```

Folosirea parametrului **secondaries** previne anunțarea adreselor secundare de pe acea interfață.

Pentru exemplificarea folosirii acestor comenzi, se va considera următorul exemplu de configurare. Se va folosi topologia din figura 5-18, şi se va configura OSPF cu o singură zonă.

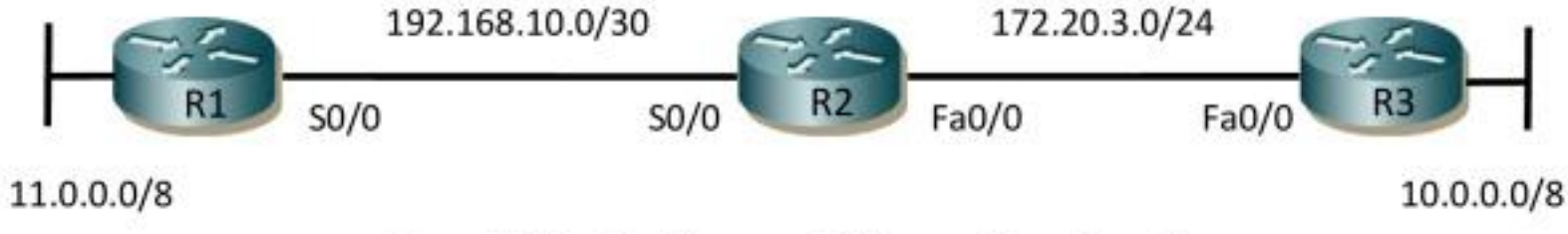

Figura 5-18: : Configurare OSPF cu o singură zonă

Ruterul R1 va trebui configurat în felul următor:

```
R1(config)# router ospf 1
R1(config-router)# network 11.0.0.0 0.255.255.255 area 0
R1(config-router)# network 192.168.10.0 0.0.0.3 area 0
```

Pentru ruterul R2 vor trebui introduse umătoarele comenzi:

```
R2(config)# router ospf 2
R2(config-router)# network 192.168.10.0 0.0.0.3 area 0
R2(config-router)# network 172.20.3.0 0.0.0.255 area 0
```

Pentru ruterul R3 următoarele comenzi:

```
R3(config)# router ospf 50
R3(config-router)# network 172.20.3.0 0.0.0.255 area 0
R3(config-router)# network 10.0.0.0 0.255.255.255 area 0
```

Se va folosi topologia din figura 5-19, pentru a exemplifica configurarea OSPF pentru mai multe zone.

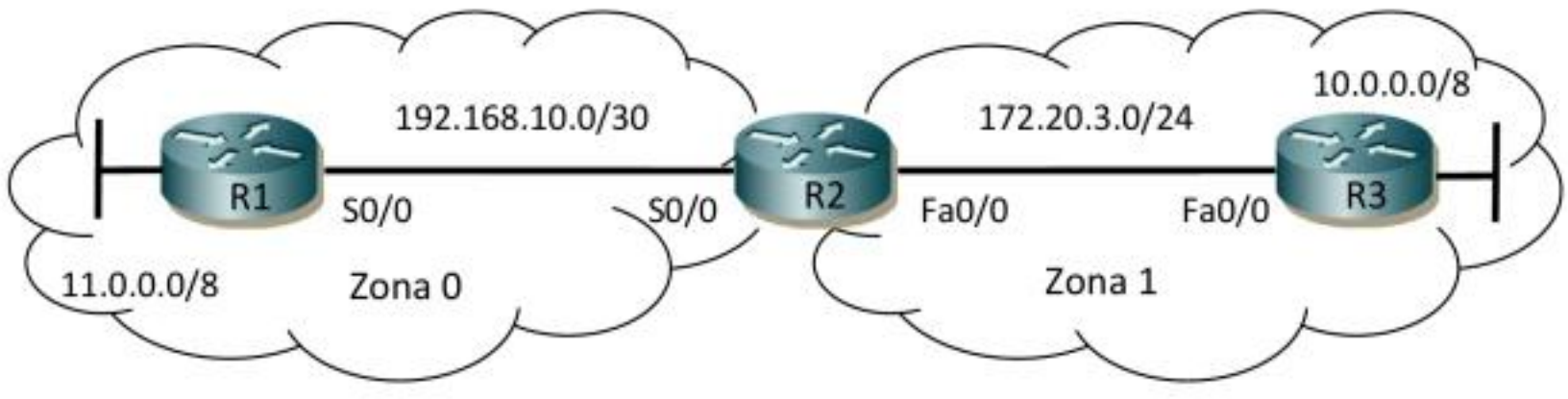

Figura 5-19: Configurare OSPF cu o mai multe zone

Ruterul R1 va trebui configurat în felul următor:

```
R1(config)# router ospf 1
R1(config-router)# network 11.0.0.0 0.255.255.255 area 0
R1(config-router)# network 192.168.10.0 0.0.0.3 area 0
```

Pentru ruterul R2 vor trebui introduse umătoarele comenzi:

```
R2(config)# router ospf 2
R2(config-router)# network 192.168.10.0 0.0.0.3 area 0
R2(config-router)# network 172.20.3.0 0.0.0.255 area 1
```

Pentru ruterul R3 următoarele comenzi:

```
R3(config)# router ospf 50
R3(config-router)# network 172.20.3.0 0.0.0.255 area 1
R3(config-router)# network 10.0.0.0 0.255.255.255 area 1
```

Există două metode de modificare a costului unei legături:

1. Prin modificarea lățimii de bandă declarate, cu consecințe asupra tuturor protocoalelor ce folosesc această informație:

```
R(config-if)# bandwidth valoare
```

2. Prin stabilirea explicită a costului, ceea ce afectează doar metrica OSPF:

```
R(config-if)# ip ospf cost valoare
```

Atunci când porneşte, procesul OSPF alege un identificator al ruterului. Acest identificator poate fi:

- cea mai mare adresă IP configurată pe una dintre interfețele active
- cea mai mare adresă IP configurată pe o interfață de loopback
- un identificator configurat manual

Dacă este configurat un identificator, are prioritate la alegere în fața adreselor IP de pe interfețele active sau de loopback. Daca nu este configurat un identificator, adresa IP a interfeței de loopback are prioritate în fața adreselor de pe interfețele active, chiar daca au adresa IP mai mare. În cazul în care nu este configurată nici o adresă pe intefața de loopback, se va alege cea mai mare adresă IP de pe una din interfețele active.

Comanda prin care se configurează identificatorul ruterului este următoarea:

```
R(config-router)# router-id adresa_ip
```

Identificatorul ruterului trebuie să fie o valoare pe 32 de biți cu format de adresă IP. De asemenea, trebuie să fie unic în sistemul autonom.

Dacă un alt identificator a fost deja selectat trebuie forțată reselecția prin comanda:

```
R# clear ip ospf process
```

Trebuie reținut faptul că această comandă întrerupe temporar funcționarea rețelei.

5.3.2 Verificarea funcționării OSPF

Folosind topologia din figura 5-19, se poate verifica dacă OSPF a fost configurat în mod corect folosind comenzile prezentate în continuare.

Pentru a verifica identificatorul ruterului şi alți parametri OSPF se foloseşte comanda **`show ip ospf`**:

```
R1# show ip ospf
Routing Process "ospf 1" with ID 192.168.10.1
  Supports only single TOS(TOS0) routes
  Supports opaque LSA
  SPF schedule delay 5 secs, Hold time between two SPFs 10 secs
  Minimum LSA interval 5 secs. Minimum LSA arrival 1 secs
  LSA group pacing timer 100 secs
  Interface flood pacing timer 55 msecs
  Retransmission pacing timer 100 msecs
  Number of external LSA 0. Checksum Sum 0x0
  Number of opaque AS LSA 0. Checksum Sum 0x0
  Number of DCbitless external and opaque AS LSA 0
  Number of DoNotAge external and opaque AS LSA 0
  Number of areas in this router is 2. 2 normal 0 stub 0 nssa
  External flood list length 0
     Area BACKBONE(0)
         Number of interfaces in this area is 2
         Area has message digest authentication
         SPF algorithm executed 4 times
         Area ranges are
         Number of LSA 2. Checksum Sum 0x29BEB
         Number of opaque link LSA 0. Checksum Sum 0x0
         Number of DCbitless LSA 1
         Number of indication LSA 0
         Number of DoNotAge LSA 0
         Flood list length 0
```

Pentru a verifica dacă interfețele sunt incluse în ariile OSPF se va folosi comanda **show ip ospf interface**:

```
R2# show ip ospf interface Fa0/0
FastEthernet0/0 is up, line protocol is up
  Internet Address 172.20.3.1/24, Area 0
  Process ID 2, Router ID 192.168.10.2, Network Type BROADCAST, Cost:
1
  Transmit Delay is 1 sec, State BDR, Priority 1
  Designated Router (ID) 172.20.3.2, Interface address 172.20.3.2
  Backup Designated router (ID) 192.168.10.2, Interface address
172.20.3.1
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    Hello due in 00:00:06
  Index 1/1, flood queue length 0
  Next 0x0(0)/0x0(0)
  Last flood scan length is 2, maximum is 2
  Last flood scan time is 0 msec, maximum is 4 msec
  Neighbor Count is 1, Adjacent neighbor count is 1
    Adjacent with neighbor 172.20.3.2  (Designated Router)
  Suppress hello for 0 neighbor(s)
```

Comanda anterioară afișează informații despre identificatorul procesului, identificatorul ruterului, tipul rețelei, DR și BDR, intervale de timp configurate și vecinii cu care a format o adiacență.

Ruterele care rulează OSPF nu trimit și nu primesc actualizări dacă nu s-au format adiacențe cu vecinii cu care urmează să comunice. Adiacențele formate se pot verifica folosind comanda **show ip ospf neighbor**:

```
R2# show ip ospf neighbor

Neighbor ID     Pri   State      Dead Time   Address
Interface
172.20.3.2      1     FULL/DR    00:00:36    172.20.3.2
FastEth0/0
192.168.10.1    0     FULL/-     00:00:38    192.168.10.1
Serial0/0
```

Prima intrare în tabelă reprezintă adiacența formată pe interfața FastEthernet. Starea FULL înseamnă că tabelele de topologie au fost schimbate între ruterele vecine iar DR înseamnă ca ruterul vecin este DR pentru rețeaua respectivă, de aceea și prioritatea este 1. A doua linie reprezintă adiacența cu vecinul aflat pe legătura serială. Lipsa parametrului DR/BDR/DROTHER se datorează faptului că legătura este serială, deoarece doar pe legături multi-acces se alege DR și BDR.

Tabela de topologie se poate afișa folosind comanda show **`ip ospf database`**. Se vor afișa toate LSA-urile grupate după tip pentru fiecare zonă în parte.

```
R1# show ip ospf database
OSPF Router with ID(192.168.10.1) (Process ID 1)

                Router Link States(Area 0)

 Link ID         ADV Router      Age         Seq#       Checksum Link
count
 11.0.0.1        11.0.0.1        1381        0x8000010D    0xEF60   1
 192.168.10.1    192.168.10.1    1460        0x800002FE    0xEB3D   2
 192.168.10.2    192.168.10.2    2027        0x80000090    0x875D   2

                Summary Net Link States(Area 0)

 Link ID         ADV Router      Age         Seq#       Checksum
 172.20.3.0      192.168.10.2    1323        0x8000005B    0xA8EE
 10.0.0.0        192.168.10.2    1461        0x8000005B    0x7AC1
```

Coloanele tabelului au semnificația următoare:

- **Link ID** este identificatorul LSA-ului
- **ADV Router** este ruterul care a generat acest LSA
- **Age** este varsta LSA-ului în secunde
- **Seq#** este numărul de secvență al LSA-ului. Inițial este 0x80000001 și creşte la fiecare actualizare
- **Checksum** este suma de control a pachetului LSA
- **Link count** este numarul de legături direct conectate și este folosita doar pentru LSA de tip Router. Legăturile seriale se pun ca 2 legături iar toate celelalte tipuri se pun ca o legătură.

Cele mai bune rute vor fi instalate în tabela de rutare. Pentru a afișa tabela de rutare se folosește comanda show ip route:

```
R1# show ip route ospf
O IA   10.0.0.0/8     [110/782] via 192.168.10.2,   00:00:52, Serial0/0
O IA   172.20.3.0/24 [110/74] via 192.168.10.2, 00:00:49, Serial0/0
```

Se pot observa rutele învățate de R1 prin intermediul lui R2. Notația este O IA (Inter-Area) deoarece rutele provin din altă zonă. Pentru rute din aceeași zona se folosește notația O, pentru rute externe E1 și E2, iar pentru rute externe provenite dintr-o zonă NSSA, N1 și N2.

Pentru a vizualiza infomații despre protocolul de rutare și parametri cum ar fi intervale de timp, metrici, rețele, se folosește comanda **`show ip protocols`**.

5.3.3 Configurarea unei zone Stub

Pentru a obține o zonă Stub, este necesară configurarea fiecărui ruter din zonă folosind comanda stub.

```
R(config-router)# area id_zona stub
```

Pentru a defini costul rutei default trimise în zona Stub se va folosi următoarea comandă pe ruterul ABR de la granița zonei Stub:

```
R(config-router)# area id_zona default-cost cost
```

Costul implicit al rutei este 1, iar comanda anterioară acceptă o valoare între 0 şi 16777215.

Folosind topologia din figura 5-20, se va configura zona 2 ca fiind Stub. Ruterul R4 nu va primi informaţii despre reţeaua externă, dar va putea primi rute sumarizate. O rută default va fi injectată de către ruterul R3 în zona Stub pentru a putea accesa reţeaua externă.

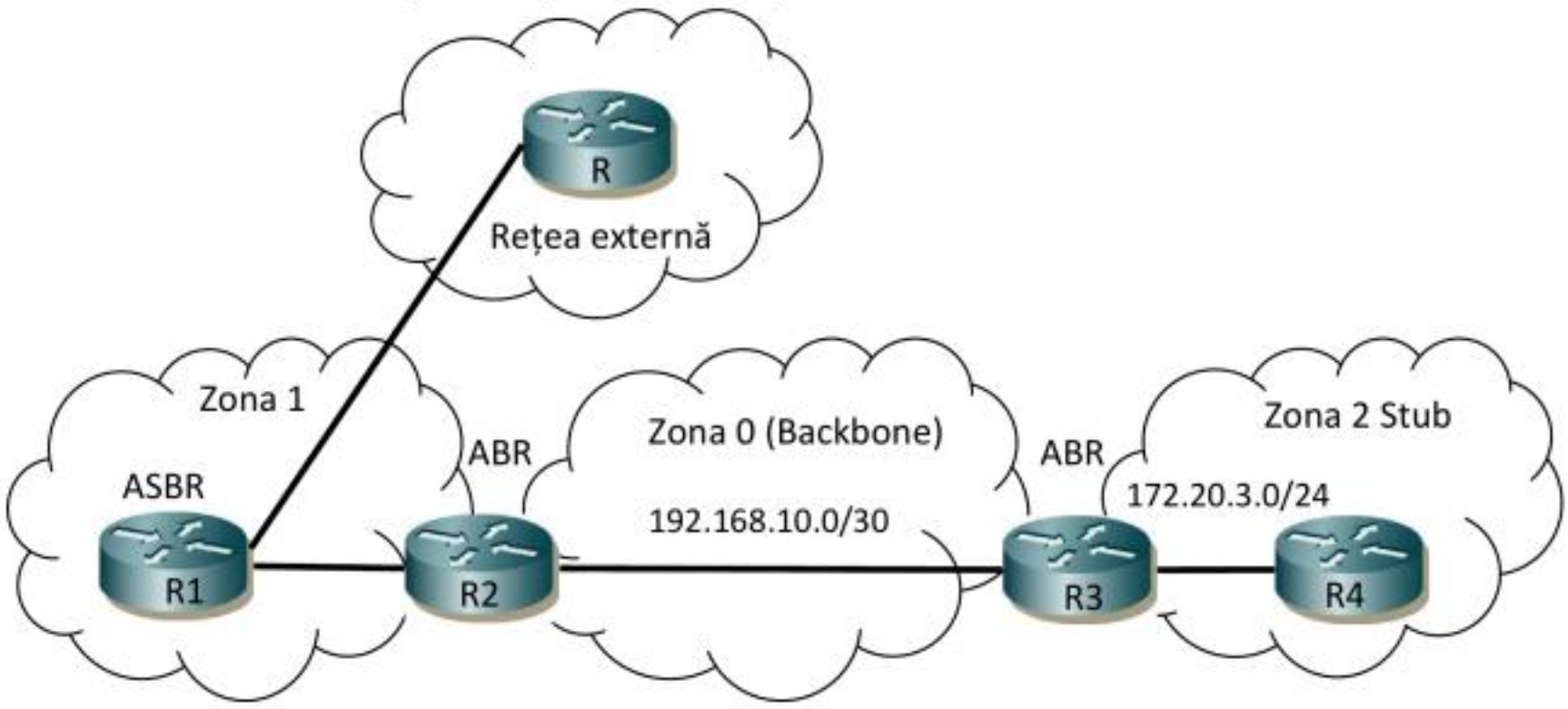

Figura 5-20: : Configurare zonă Stub

Ruterul R3 va trebui configurat în felul următor:

```
R3(config)# router ospf 1
R3(config-router)# network 192.168.10.0 0.0.0.3 area 0
R3(config-router)# network 172.20.3.0 0.0.0.255 area 2
R3(config-router)# area 2 stub
```

Pentru ruterul R4 vor trebui introduse umătoarele comenzi:

```
R4(config)# router ospf 2
R4(config-router)# network 172.20.3.0 0.0.0.255 area 2
R4(config-router)# area 2 stub
```

Zona Stub se va forma în momentul în care toate ruterele care aparţin de acea zonă au fost configurate folosind comanda **`area stub`**.

5.3.4 Configurare zonă Totally Stubby

După cum a fost prezentat în subcapitolul 5.2.1, o zonă Totally Stubby va opri pachetele LSA de tip 3, 4 şi 5. O zonă Totally Stubby va fi setată, mai întâi, prin configurarea fiecăruiruter din zonă, în afară de ABR, pentru a face parte din zona Stub:

```
R(config-router)# area id_zona stub
```

Apoi, se va configura ruterul ABR folosind opţiunea no-summary, ceea ce va opri toate pachetele LSA de tip Summary să intre în zonă:

```
R(config-router)# area id_zona stub no-summary
```

Folosind aceeaşi topologie din figura 5-20, se va configura zona 2 ca fiind Totally Stubby.

Ruterul R3 va trebui configurat în felul următor:

```
R3(config)# router ospf 1
```

```
R3(config-router)# network 192.168.10.0 0.0.0.3 area 0
R3(config-router)# network 172.20.3.0 0.0.0.255 area 2
R3(config-router)# area 2 stub no-summary
```

Pentru ruterul R4 vor trebui introduse următoarele comenzi:

```
R4(config)# router ospf 2
R4(config-router)# network 172.20.3.0 0.0.0.255 area 2
R4(config-router)# area 2 stub
```

5.3.5 Configurare zonă Not So Stubby

Noțiunea de zonă NSSA a fost prezentată în subcapitolul 5.2.1. NSSA este o zonă Stub în care este permisă prezența unui ruter ASBR. O zonă va fi definită ca fiind NSSA dacă toate ruterele din zonă vor fi configurate folosind următoarea comandă:

```
R(config-router)# area id_zona nssa [no-summary]
```

Folosind topologia din figura 5-21, se va configura zona 1 ca fiind NSSA. Ruterul R1 va redistribui rute RIP în OSPF şi va genera un LSA de tip 7. Ruterul R2 va converti LSA7 în LSA5. Ruterul 2 trebuie configurat să sumarizeze rutele externe redistribuite din RIP, astfel încât subrețelele 10.1.0.0/16, 10.2.0.0/16 şi 10.3.0.0/16 vor fi sumarizate ca 10.0.0.0/8 şi va fi trimis un singur LSA în zona 0. Pentru ca ruterul R2 să genereze o rută default de tipul *O N2 pentru zona NSSA, se va folosi opțiunea **default-information-originate** a comenzii **area nssa**.

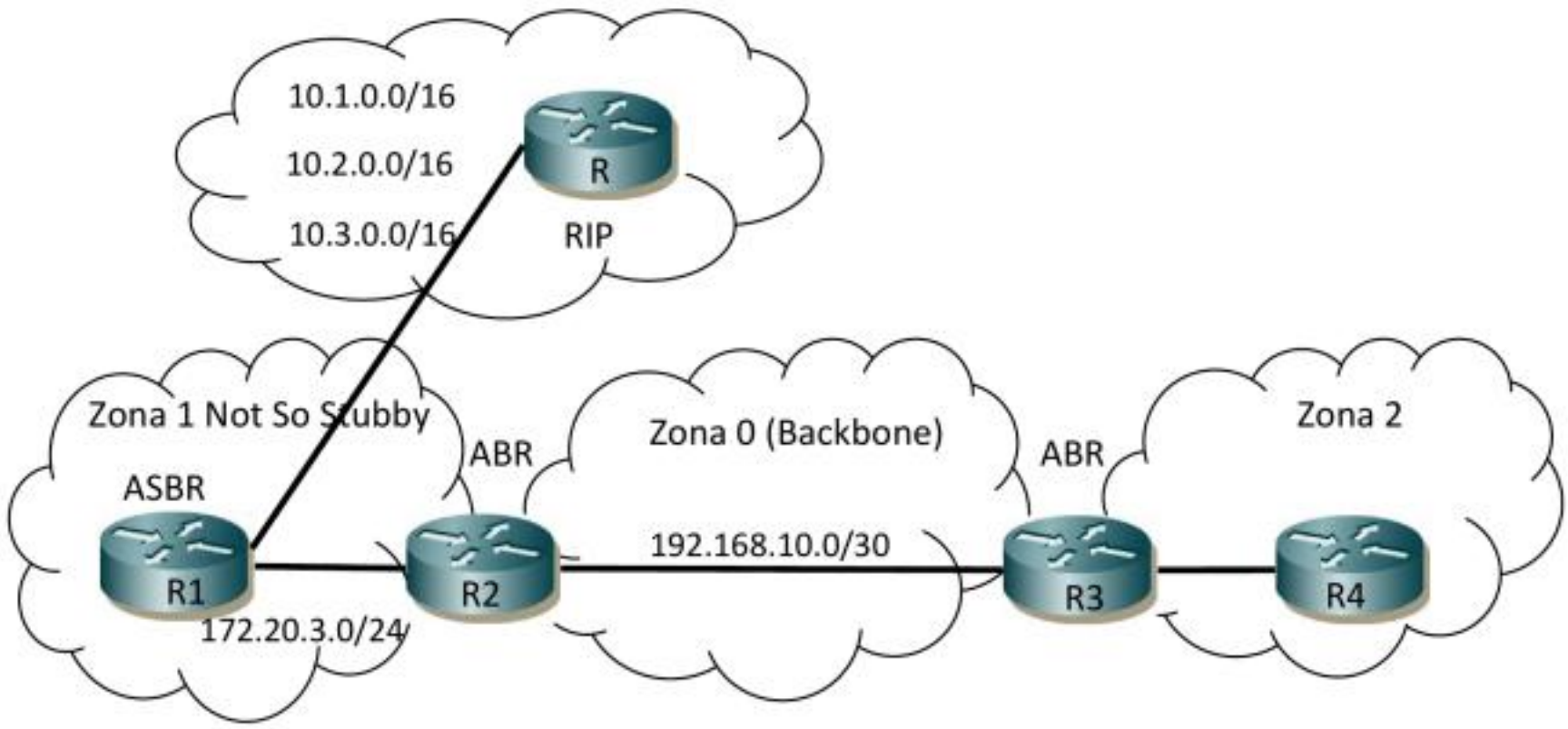

Figura 5-21: Zona Not So Stubby

Ruterul R2 va trebui configurat în felul următor:

```
R2(config)# router ospf 1
R2(config-router)# summary-address 10.0.0.0 255.0.0.0
R2(config-router)# network 192.168.10.0 0.0.0.3 area 0
R2(config-router)# network 172.20.3.0 0.0.0.255 area 1
R2(config-router)# area 1 nssa default-information-originate
```

Pentru ruterul R1 vor trebui introduse umătoarele comenzi:

```
R1(config)# router ospf 2
R1(config-router)# redistribution rip subnets
R1(config-router)# default metric 150
R1(config-router)# network 172.20.3.0 0.0.0.255 area 1
R1(config-router)# area 1 nssa
```

Pentru ca zona 1 să fie NSSA Totally Stubby, se va folosi opțiunea **no-summary** a comenzii **area nssa**. Astfel se va genera automat şi o rută default *O N2 pentru zona NSSA, deci nu mai este necesară folosirea opțiunii **default-information-originate**.

Folosind comanda următoare se vor putea vizualiza zonele normale, Stub şi NSSA:

```
R# show ip ospf
```

Următoarea comandă afişează detalii despre LSA de tip 7 din tabela de topologie:

```
R# show ip ospf database nssa-external
```

Rutele externe redistribuite de un ASBR din cadrul unei zone NSSA se vor vizualiza folosind comanda următoare:

```
R# show ip route
```

5.3.6 Configurarea sumarizării

OSPF nu realizează sumarizarea automată a rutelor la nivel de clasă. Comenzile necesare pentru a configura sumarizarea sunt diferite în funcție de tipul de sumarizare. Tipurile de sumarizări au fost definite în capitolul 5.2.2.

Sumarizarea între zone se realizează pe ABR folosindu-se comanda area range care va determina ruterul să sumarizeze rutele dintr-o anumită zonă înainte de a le injecta în zona de backbone ca LSA de tip 3.

```
R(config-router)# area id_zona range adresa masca [advertize | not-
advertize] [cost cost]
```

Zona specificată în comanda anterioară este zona din care provin rutele care vor fi sumarizate. Opțiunea **not-advertize** va opri generarea pachetelor LSA de tip 3, fapt ce va determina ascunderea rețelelor ce fac parte din intervalul specificat de restul zonelor. Opțiunea **cost** este folosită pentru a configura manual metrica rutei sumarizate care va fi folosită în calculul STP. Costul specificat trebuie să fie mai mare decât 0 şi mai mic decât 16777215.

Sumarizarea rutelor externe se realizează pe ASBR înainte de a fi injectate în OSPF ca LSA de tip 5. Se va folosi următoarea comandă:

```
R(config-router)# summary-address adresa masca [not-advertize] [tag
eticheta]
```

Opțiunea **not-advertize** se va folosi pentru a opri redistribuirea intervalului de adrese specificat. Eticheta va fi folosită pentru a controla redistibuția folosint route maps.

Se va folosi topologia din figura 5-22. Se va configura sumarizare pe cele două rutere ABR. Intervalul de adrese de la 172.20.32.0/24 până la 172.20.63.0/24 va fi sumarizat în intervalul 172.20.32.0/19. Intervalul de adrese de la 172.20.64.0/24 până la 172.20.95.0/24 va fi sumarizat în intervalul 172.20.64.0/19. Intervalul de adrese de la 172.20.96.0/24 până la 172.20.127.0/24 va fi sumarizat în intervalul 172.20.64.0/19.

Pentru ruterul R1 vor trebui introduse umătoarele comenzi:

```
R1(config)# router ospf 2
R1(config-router)# area 1 range 192.168.32.0 225.255.224.0
R1(config-router)# area 0 range 192.168.96.0 255.255.224.0
```

Figura 5-22: Sumarizarea pe ABR

Ruterul R2 va trebui configurat în felul următor:

```
R2(config)# router ospf 1
R2(config-router)# area 0 range 192.168.96.0 255.255.224.0
R2(config-router)# area 2 range 192.168.64.0 255.255.224.0
```

Folosind topologia din figura 5-23, se va configura sumarizarea rutelor externe pe ABSR. Rutele externe primite prin RIP vor fi redistribuite în OSPF pe ruterul R1. Intervalul de adrese de la 172.20.32.0/24 până la 172.20.63.0/24 va fi sumarizat în intervalul 172.20.32.0/19.

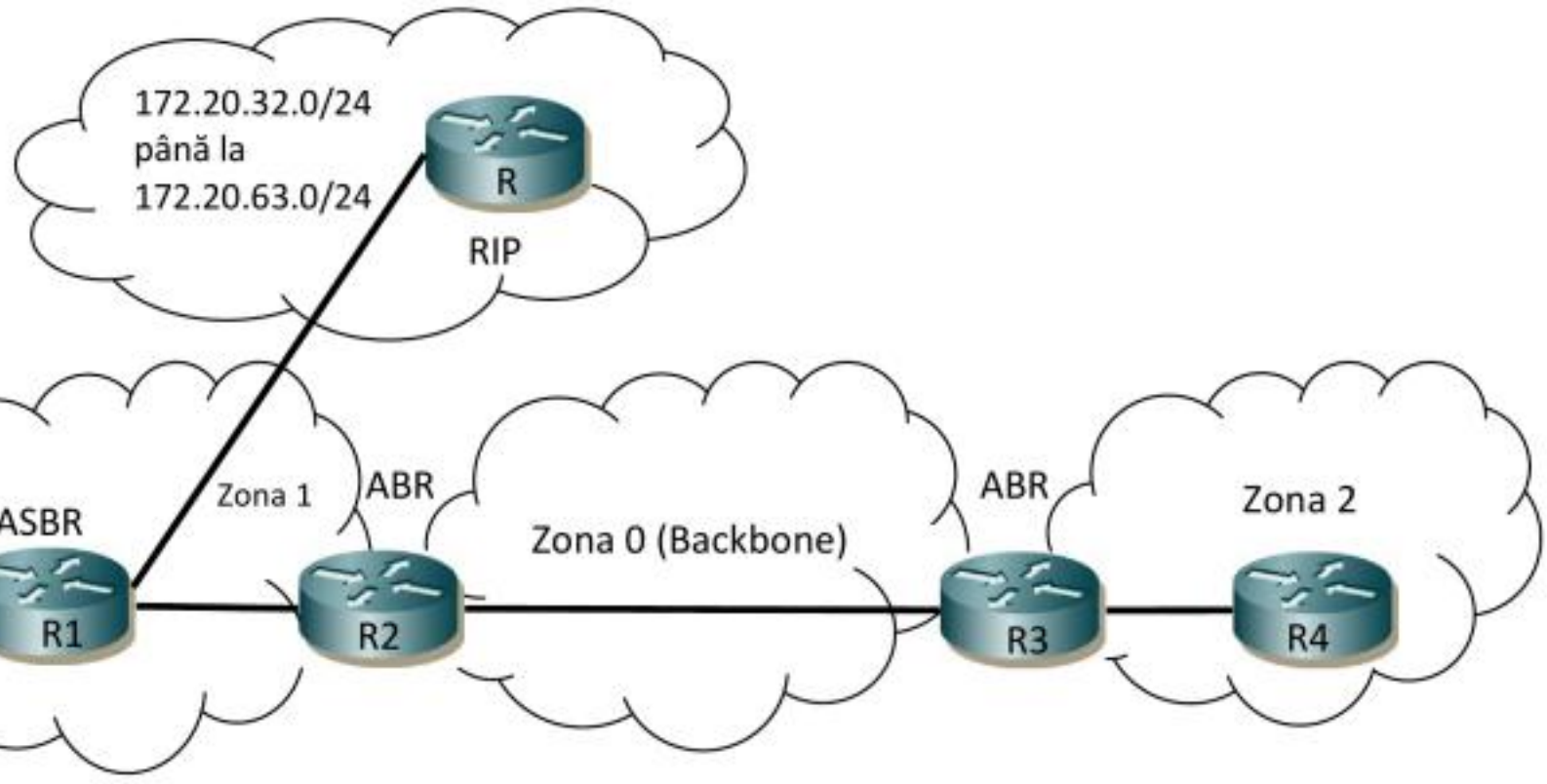

Figura 5-23: Sumarizarea rutelor externe pe ASBR

Pentru ruterul R1 vor trebui introduse umătoarele comenzi:

```
R1(config)# router ospf 2
R1(config-router)# redistribution rip subnets
R1(config-router)# default metric 150
R1(config-router)# summary-address 172.20.32.0 255.255.224.0
```

Dacă există o singură ieşire din sistemul autonom, în locul redistribuirii rutelor externe în OSPF, se poate injecta o singură rută default. Pentru a genera o rută default se va folosi comanda:

```
R(config-router)# default-information originate [always] [metric
metrica] [metric-type tip] [route-map nume]
```

În mod normal această comandă folosită fără parametri va determina anunțarea unei rute default doar dacă pe ruterul respectiv există deja o ruta default configurată. Folosirea opțiunii **always** a comenzii **default-information originate**, va determina anunțarea unei rute default fără ca aceasta să existe în mod explicit în tabela de rutare.

Metrica va specifica costul rutei default, care în mod implicit este 1. Tipul metricii a fost definit în subcapitolul 5.1.6 şi poate fi de tipul E1 sau E2. Se poate specifica un nume de route map.

O rută default va apărea în tabela de topologie ca un LSA extern de tipul 5.

Se va folosi topologia din figura 5-24 pentru a se exemplifica configurarea rutelor default.

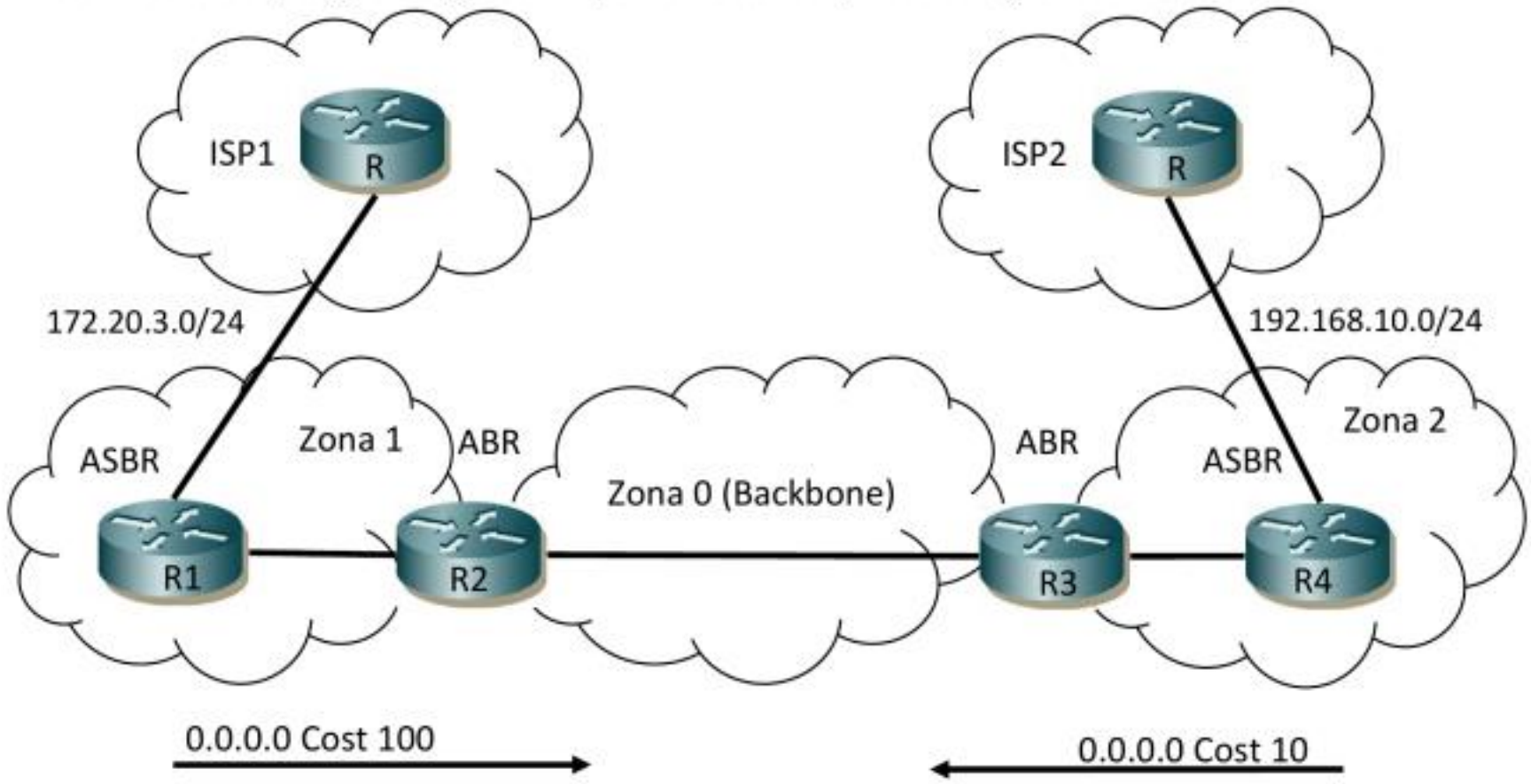

Figura 5-24: Rute default

Pe ruterul R1 se vor introduce următoarele comenzi:

```
R1(config)# ip route 0.0.0.0 172.20.3.1
R1(config)# router ospf 1
R1(config-router)# default-information originate cost 100
```

Iar pe ruterul R4 următoarele comenzi:

```
R4(config)# ip route 0.0.0.0 192.168.10.1
R4(config)# router ospf 2
R4(config-router)# default-information originate cost 10
```

Va fi preferată calea către ISP2 deoarece metrica specificată pentru ruta default prin ISP2 este mai mică decât ruta default prin ISP1.

5.3.7 Configurarea autentificării

Pentru a configura OSPF să folosească o parolă simplă pentru a autentifica ruterele vecine trebuie parcurşi următorii paşi:

1. Configurarea unei parole care sa fie folosită în comunicația cu ruterele vecine. Comanda trebuie introdusă pentru fiecare interfață în parte, astfel o parolă distinctă poate fi configurată pentru fiecare interfață. Parola trebuie sa corespundă pentru interfețele de pe rutere care formează o legătură.

```
R(config-if)# ip ospf authentication-key parola
```

Parola trebuie să fie un şir de maxim 8 caractere.

2. Specificarea tipului autentificării.

```
R(config-if)# ip ospf authentication [message-digest | null]
```

Comanda anterioră folosită fără nici o opțiune va determina autentificarea folosind parola simpla, trimisă în clar. Pentru a se folosi autentificarea prin MD5 se va specifica opțiunea **message-digest iar** pentru a dezactiva autentificarea se foloseşte opțiunea **null.**

Se va folosit topologia din figura 5-24, pentru a configura autentificarea prin parola simplă între ruterele R1 şi R2.

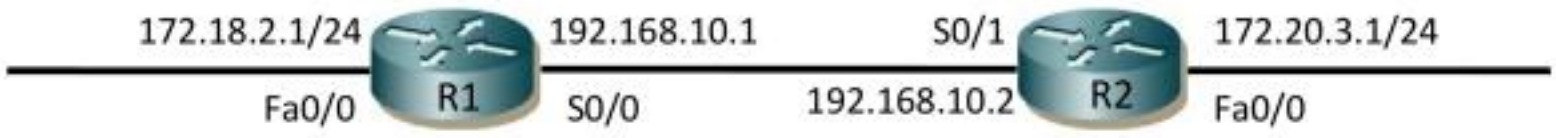

Figura 5-25: Autentificarea în OSPF

Pentru a configura ruterul R1 trebuie introduse următoarele comenzi:

```
R1(config)# interface S0/0
R1(config-if)# ip ospf authentication-key parola
R1(config-if)# ip ospf autentication
```

Pentru a configura ruterul R2 trebuie introduse următoarele comenzi:

```
R2(config)# interface S0/1
R2(config-if)# ip ospf authentication-key parola
R2(config-if)# ip ospf autentication
```

Pentru configura autentificarea MD5, o cheie şi un identificator de cheie trebuie specificate pe fiecare ruter. Următorii paşi trebuie parcurşi:

1. Asignarea unei chei şi a unui identificator de cheie pe fiecare interfață. O cheie diferită poate fi configurată pe fiecare interfață în parte. Cheia trebuie sa corespundă pentru interfețele de pe rutere care formează o legătură. Acelaşi identificator de cheie pe ruterul vecin trebuie să aibă ataşată aceeaşi cheie pentru a se realiza autentificarea. Identificatorii de cheie permit tranzițiile între chei fără a opri comunicația pe acea legătură.

```
R(config-if)# ip ospf message-digest-key id_cheie md5 cheie
```

Identificatorul cheii trebuie să fie un număr între 1 şi 255 iar cheia poate să aibă până la 16 caractere. Cheia va fi folosită pentru a genera un rezumat al fiecărui pachet OSPF. Rezumatul va fi ataşat la pachet.

2. Specificarea tipului autentificării.

```
R(config-if)# ip ospf authentication message-digest
```

O comandă alternativă care configurează tipul autentificării la nivel de zonă este următoarea:

```
R(config-router)# area id_zona authentication [message-digest]
```

Se va folosi topologia din figura 5-24, pentru a configura autentificarea prin MD5 între ruterele R1 şi R2.

Pentru a configura ruterul R1 trebuie introduse următoarele comenzi:

```
R1(config)# interface S0/0
R1(config-if)# ip ospf message-digest-key 1 md5 secret
R1(config-if)# ip ospf autentication message-digest
```

Pentru a configura ruterul R2 trebuie introduse următoarele comenzi:

```
R2(config)# interface S0/1
R2(config-if)# ip ospf message-digest-key 1 md5 secret
R2(config-if)# ip ospf autentication message-digest
```

În cazul în care autentificare are loc cu succes, se va stabili adiacența între cele două rutere, şi se vor trimite actualizări.

Se va verifica dacă ruterele au stabilit o adiacență folosind comanda următoare:

```
R1# show ip ospf neighbor

Neighbor ID     Pri   State      Dead Time    Address
Interface
192.168.10.2    1     FULL/-     00:00:36     192.168.10.2
Serial0/0
```

Pentru a verifica trimiterea de actualizări se poate determinadacă ruta către 172.20.3.0/24 se află în tabela de rutare folosind comanda show ip route sau se poate da ping către adresa 172.20.3.1.

Pentru identificarea problemelor în legătură cu autentificarea în OSPF se va folosi comanda următoare:

```
R1# debug ip ospf adj

00:59:42: OSPF: Send with youngest Key 1
00:59:42: OSPF: 2 Way Communication to 192.168.10.2 on Serial0/0,
  state 2WAY
00:59:42: OSPF: Send DBD to 192.168.10.2 on Serial0/0 seq 0x2125 opt
0x42 flag 0x7len 32
00:59:42: OSPF: Send with youngest Key 1
00:59:42: OSPF: Rcv DBD from 192.168.10.2 on Serial0/0 seq 0x11F3 opt
0x42 flag 0x7 len 32 mtu 1500 state EXSTART
00:59:42: OSPF: First DBD and we are not SLAVE
00:59:42: OSPF: Rcv DBD from 192.168.10.2 on Serial0/0 seq 0x2125 opt
0x42 flag 0x2 len 72 mtu 1500 state EXSTART
00:59:42: OSPF: NBR Negotiation Done. We are the MASTER
00:59:42: OSPF: Send DBD to 192.168.10.2 on Serial0/0 seq 0x2126 opt
0x42 flag 0x3 len 72
00:59:42: OSPF: Send with youngest Key 1
00:59:42: OSPF: Send with youngest Key 1
00:59:42: OSPF: Database request to 192.168.10.2
00:59:42: OSPF: sent LS REQ packet to 192.16.10.2, length 12
00:59:42: OSPF: Rcv DBD from 192.168.10.2 on Serial0/0 seq 0x2126 opt
0x42 flag 0x0 len 32 mtu 1500 state EXCHANGE
00:59:42: OSPF: Send DBD to 192.168.10.2 on Serial0/0 seq 0x2127 opt
0x42 flag 0x1len 32
00:59:42: OSPF: Send with youngest Key 1
00:59:42: OSPF: Send with youngest Key 1
00:59:42: OSPF: Rcv DBD from 192.168.10.2 on Serial0/0 seq 0x2127 opt
0x42 flag 0x0 len 32 mtu 1500 state EXCHANGE
00:59:42: OSPF: Exchange Done with 192.168.10.2 on Serial0/0
00:59:42: OSPF: Synchronized with 192.168.10.2 on Serial0/0, state
FULL
00:59:42: %OSPF-5-ADJCHG: Process 10, Nbr 192.168.10.2 on Serial0/0
from
    LOADING to FULL, Loading Done
00:59:43: OSPF: Build router LSA for area 0, router ID 172.18.2.1,
  seq 0x80000010
00:59:43: OSPF: Send with youngest Key 1
00:59:45: OSPF: Send with youngest Key 1
```

Se poate observa că autentificarea s-a realizat cu succes şi s-a format o adiacență între cei doi vecini.

5.3.8 Configurarea legăturilor virtuale

O legătură virtuală se va crea folosind comanda:

```
R(config-router)# area id_zona virtual-link id_ruter [authentication
[message-digest | null]] [hello-interval secunde] [retransmit-interval
secunde] [transmit-delay secunde] [dead-interval secunde]
[[authentication-key parola] | [message-digest-key id_cheie md5 cheie]]
```

Zona specificată în comanda anterioară este zona de tranzit prin care trece legătura virtuală. Identificatorul ruterului se referă la identificatorul vecinului din capătul celălalt al legăturii virtuale. Parametri de autentificare se folosesc în mod asemănător cu cei folosiți impreună cu comanda **ip ospf**, descrişi în capitolul anterior. Opțiunea hello-interval specifică numărul de secunde între două mesaje de tip Hello transmise pe legătura virtuală, valoarea implicită fiind de 10 secunde. Retransmision-interval se referă la intervalul de timp între două retransmisii de LSA, valoarea implicită fiind de 5 secunde. Transmit-delay este timpul estimat pentru a transmite un pachet de tip LSU pe interfață, valoarea implicită fiind de 1 secundă. Dead-interval este intervalul de timp care trebuie să treacă fără să se primească nici un pachet de tip Hello, pentru ca vecinul să fie considerat inaccesibil.

Folosind topologia din figura 5-25, se va configura o legătură virtuală între ruterele R2 şi R3 prin zona de tranzit 1 care leagă doua zone de backbone.

Figura 5-26: Legături virtuale

Pe ruterul R2 trebuie introduse următoarele comenzi:

```
R2(config)# router ospf 1
R2(config-router)# network 10.0.0.0 0.255.255.255 area 0
R2(config-router)# network 192.168.10.0 0.0.0.255 area 1
R2(config-router)# area 1 virtual-link 10.2.2.2
```

Iar pe ruterul R3 următoarele comenzi:

```
R3(config)# router ospf 2
R3(config-router)# network 10.0.0.0 0.255.255.255 area 0
R3(config-router)# network 192.168.10.0 0.0.0.255 area 1
R3(config-router)# area 1 virtual-link 10.1.1.1
```

Pe fiecare ruter trebuie să se specifice identificatorul ruterului vecin.

Pentru a verifica dacă legătura virtuală functionează în mod corect se foloseşte comanda:

```
R2# show ip ospf virtual-links

Virtual Link OSPF_VL0 to router 10.2.2.2 is up
Transit area 0.0.0.1, via interface Serial0/0, Cost of using 781
Transmit Delay is 1 sec, State POINT_TO_POINT
Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
Hello due in 0:00:08
Adjacency State FULL
Index 1/2, retransmission queue length 0, number of retransmission 1
```

```
    First 0x0(0)/0x0(0) Next 0x0(0)/0x0(0)
    Last retransmission scan length is 1, maximum is 1
    Last retransmission scan time is 0 msec, maximum is 0 msec
```

Prima linie din output specifică dacă legătura virtuală este activă sau nu. Apoi sunt specificate informații despre aria de tranzit, interfața spre vecin, costul legăturii, intervalele configurate și starea adiacenței.

Folosind comanda show ip ospf neighbor se poate verifica dacă s-a stabilit o adiacență cu vecinul de la celălalt capăt al legăturii virtuale:

```
    R2# show ip ospf neighbor

    Neighbor ID     Pri    State      Dead Time    Address
Interface
    10.10.10.1      1      FULL/DR    00:00:35     10.1.1.2
FastEth0/0
    10.2.2.2        0      FULL/-     -            192.168.10.2   OSPF_VL0
    10.2.2.2        0      FULL/-     00:00:36     192.168.10.2
Serial0/0
```

Se observă că s-a stabilit o adiacență cu ruterul care are identificatorul 10.1.2.1 prin legătura virtuală numită OSPF_VL0. De asemenea s-a stabilit o adiacență cu același ruter prin legătura directă.

Se poate verifica dacă ruterele schimbă LSA-uri folosind următoarea comandă:

```
    R2# show ip ospf database router 10.1.2.1
            OSPF Router with ID (10.1.1.1) (Process ID 1)

                    Router Link States (Area 0)

    Routing bit set on this LSA
    LS age: 1 (DoNotAge)
    Options: (No TOS-capability, DC)
    LS Type: Router Links
    Link State ID: 10.2.2.2
    Advertising Router: 10.2.2.2
    LS Seq Number: 80000003
    Checksum: 0xD5DF
    Length: 48
    Area Border Router
    Number of Links: 2

        Link connected to: a Virtual Link
         (Link ID) Neighboring Router ID: 10.1.1.1
         (Link Data) Router Interface address: 192.168.10.2
          Number of TOS metrics: 0
           TOS 0 Metrics: 781

        Link connected to: a Transit Network
         (Link ID) Network/subnet number: 10.1.2.2
         (Link Data) Router Interface address: 10.1.2.2
          Number of TOS metrics: 0
           TOS 0 Metrics: 1

                    Router Link States (Area 1)

    Routing bit set on this LSA
    LS age: 1688
    Options: (No TOS-capability, DC)
    LS Type: Router Links
    Link State ID: 10.2.2.2
    Advertising Router: 10.2.2.2
    LS Seq Number: 80000008
    Checksum: 0xCC81
    Length: 48
```

```
Area Border Router
Virtual Link Endpoint
Number of Links: 2

    Link connected to: another Router (point-to-point)
     (Link ID) Neighboring Router ID: 10.1.1.1
     (Link Data) Router Interface address: 192.168.10.2
      Number of TOS metrics: 0
       TOS 0 Metrics: 781

    Link connected to: a Stub Network
     (Link ID) Network/subnet number: 192.168.10.0
     (Link Data) Network Mask : 255.255.255.0
      Number of TOS metrics: 0
       TOS 0 Metrics: 781
```

5.4 Studiu de caz

5.4.1 Topologie

Pentru scenariul de mai jos, compania Linkin'State decide implementarea protocolului OSPF în rețeaua sa. Compania dorește observarea rețelei în cadrul OSPF Single-area, după care va converti topologia într-una Multi-area.

Testarea capabilităților protocolului OSPF se va face pe o topologie de 5 rutere cu două segmente multiacces FastEthernet și două segmente point-to-point cu linii seriale.

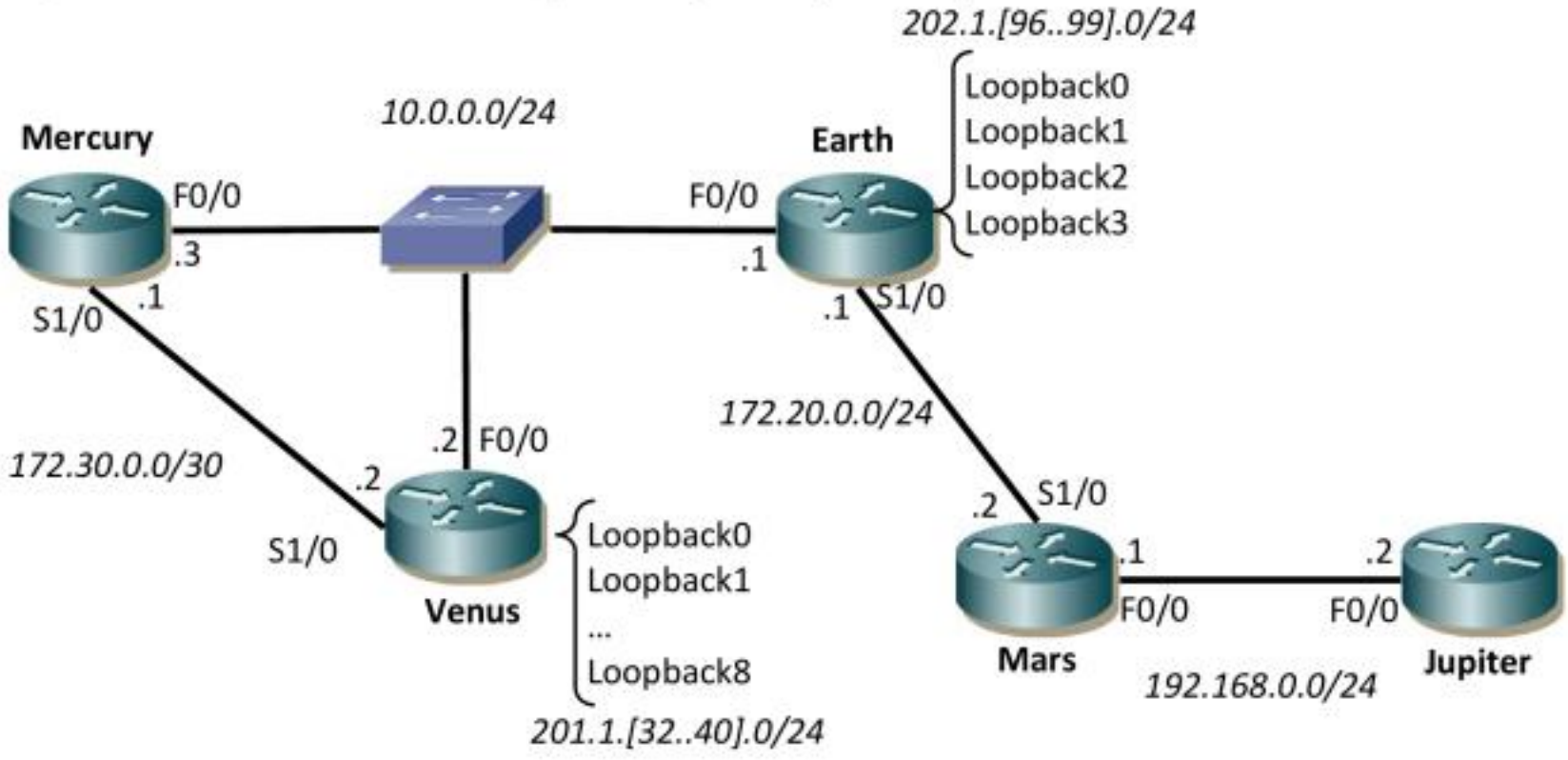

Figura 5-27: Topologie studiu de caz

Topologia de mai sus prezintă următoarea schemă de adresare:

Rețeaua switched dintre Earth, Venus și Mercury: 10.0.0.0/24 cu adresele:

Earth FastEthernet 0/0: 10.0.0.1/24

Venus FastEthernet 0/0: 10.0.0.2/24

Mercury FastEthernet 0/0: 10.0.0.3/24

Legătura point-to-point dintre Mercury și Venus: 172.30.0.0/30

Mercury Serial 1/0: 172.30.0.1/30

Venus Serial 1/0: 172.30.0.2/30

Legătura point-to-point dintre Earth și Mars: 172.20.0.0/30

Earth Serial 1/0: 172.20.0.1/30

Mars Serial 1/0: 172.20.0.2/30

Legătura FastEthernet dintre Mars și Jupiter: 192.168.0.0/24

Mars FastEthernet 0/0: 192.168.0.1/24

Jupiter FastEthernet 0/0: 192.168.0.2/24

Interfețe loopback configurate pe ruterele Venus și Earth, ce simulează alte rețele ale topologiei și a căror schemă de adresare este descrisă în output-urile următoare.

5.4.2 Configurări OSPF Single-area

Se verifică schema de adresare și configurarea interfețelor corespunzătoare:

```
Mercury#show ip interface brief
Interface              IP-Address      OK? Method Status         Protocol
FastEthernet0/0        10.0.0.3        YES manual up                 up
Serial1/0              172.30.0.1      YES manual up                 up

Venus#show ip interface brief
Interface              IP-Address      OK? Method Status         Protocol
FastEthernet0/0        10.0.0.2        YES manual up                 up
Serial1/0              172.30.0.2      YES manual up                 up
Loopback0              201.1.32.1      YES manual up                 up
Loopback1              201.1.33.1      YES manual up                 up
Loopback2              201.1.34.1      YES manual up                 up
Loopback3              201.1.35.1      YES manual up                 up
Loopback4              201.1.36.1      YES manual up                 up
Loopback5              201.1.37.1      YES manual up                 up
Loopback6              201.1.38.1      YES manual up                 up
Loopback7              201.1.39.1      YES manual up                 up
Loopback8              201.1.40.1      YES manual up                 up

Earth#show ip interface brief
Interface              IP-Address      OK? Method Status         Protocol
FastEthernet0/0        10.0.0.1        YES manual up                 up
Serial1/0              172.20.0.1      YES manual up                 up
Loopback0              202.1.96.1      YES manual up                 up
Loopback1              202.1.97.1      YES manual up                 up
Loopback2              202.1.98.1      YES manual up                 up
Loopback3              202.1.99.1      YES manual up                 up

Mars#show ip interface brief
Interface              IP-Address      OK? Method Status         Protocol
FastEthernet0/0        192.168.0.1     YES manual up                 up
Serial1/0              172.20.0.2      YES manual up                 up

Jupiter#show ip interface brief
Interface              IP-Address      OK? Method Status         Protocol
FastEthernet0/0        192.168.0.2     YES manual up                 up
```

Pentru fiecare ruter în parte, se pornește procesul OSPF și se introduc rețelele direct conectate. Interfețele membre ale acestor rețele vor participa în procesul de OSPF respectiv, vor realiza adiacențe cu alți vecini prin rețelele respective și vor schimba update-uri de rutare.

OSPF poate rula o multitudine de procese independente. Ruterele adiacente nu necesită același număr de proces configurat pentru a putea comunica. Comenzile de mai jos exemplifică pornirea procesului OSPF pentru Mercury și Earth:

```
Mercury(config)#router ospf 1
Mercury(config-router)#network 10.0.0.3 0.0.0.255 area 0
Mercury(config-router)#network 172.30.0.0 0.0.0.3 area 0

Earth(config)#router ospf 1
Earth(config-router)#network 10.0.0.0 0.0.0.255 area 0
```

```
Earth(config-router)#network 172.20.0.0 0.0.0.3 area 0
```

Pentru a include în OSPF toate interfețele loopback configurate pe Venus și Earth se pot folosi comenzi `network` mai generice:

```
Venus(config-router)#net 201.1.0.0 0.0.255.255 area 0
Earth(config-router)#net 202.1.0.0 0.0.255.255 area 0
```

Se verifică tabela de rutare pe Jupiter:

```
Jupiter#sh ip route
[...]

202.1.96.0/32 is subnetted, 1 subnets
O       202.1.96.1 [110/66] via 192.168.0.1, 00:01:29, FastEthernet0/0
     201.1.39.0/32 is subnetted, 1 subnets
O       201.1.39.1 [110/67] via 192.168.0.1, 00:01:29, FastEthernet0/0
     202.1.97.0/32 is subnetted, 1 subnets
O       202.1.97.1 [110/66] via 192.168.0.1, 00:01:29, FastEthernet0/0
     201.1.38.0/32 is subnetted, 1 subnets
O       201.1.38.1 [110/67] via 192.168.0.1, 00:01:29, FastEthernet0/0
     202.1.98.0/32 is subnetted, 1 subnets
O       202.1.98.1 [110/66] via 192.168.0.1, 00:01:29, FastEthernet0/0
     201.1.37.0/32 is subnetted, 1 subnets
O       201.1.37.1 [110/67] via 192.168.0.1, 00:01:29, FastEthernet0/0
     202.1.99.0/32 is subnetted, 1 subnets
O       202.1.99.1 [110/66] via 192.168.0.1, 00:01:29, FastEthernet0/0
     201.1.36.0/32 is subnetted, 1 subnets
O       201.1.36.1 [110/67] via 192.168.0.1, 00:01:29, FastEthernet0/0
     201.1.35.0/32 is subnetted, 1 subnets
O       201.1.35.1 [110/67] via 192.168.0.1, 00:01:29, FastEthernet0/0
     201.1.34.0/32 is subnetted, 1 subnets
O       201.1.34.1 [110/67] via 192.168.0.1, 00:01:29, FastEthernet0/0
     201.1.33.0/32 is subnetted, 1 subnets
O       201.1.33.1 [110/67] via 192.168.0.1, 00:01:29, FastEthernet0/0
     172.20.0.0/30 is subnetted, 1 subnets
O       172.20.0.0 [110/65] via 192.168.0.1, 00:01:29, FastEthernet0/0
     172.30.0.0/30 is subnetted, 1 subnets
O       172.30.0.0 [110/130] via 192.168.0.1, 00:01:29, FastEthernet0/0
     201.1.32.0/32 is subnetted, 1 subnets
O       201.1.32.1 [110/67] via 192.168.0.1, 00:01:29, FastEthernet0/0
     10.0.0.0/24 is subnetted, 1 subnets
O       10.0.0.0 [110/66] via 192.168.0.1, 00:01:29, FastEthernet0/0
C    192.168.0.0/24 is directly connected, FastEthernet0/0
     201.1.40.0/32 is subnetted, 1 subnets
O       201.1.40.1 [110/67] via 192.168.0.1, 00:01:29, FastEthernet0/0
```

Se observă că toate rețelele incluse în OSPF, alături de rețelele direct conectate, au fost inserate în tabela de rutare a lui Jupiter.

5.4.2.1 Adiacențe, DR, BDR

Identificatorii ruterelor dintr-o rețea OSPF sunt setați automat la valoarea celei mai mari adrese IP configurate pe o interfață activă (loopback, dacă există cel puțin una). Configurarea locală a protocoalelor de rutare precum și Router ID -ul pentru procesul de OSPF pot fi vizualizate prin comanda `show ip protocols`. Spre exemplu, pentru ruterul Mercury:

```
Mercury#show ip protocols
Routing Protocol is "ospf 1"
  Outgoing update filter list for all interfaces is not set
  Incoming update filter list for all interfaces is not set
  Router ID 172.30.0.1
  Number of areas in this router is 1. 1 normal 0 stub 0 nssa
  Maximum path: 4
  Routing for Networks:
    10.0.0.0 0.0.0.255 area 0
```

```
    172.30.0.0 0.0.0.3 area 0
  Routing Information Sources:
    Gateway         Distance      Last Update
    192.168.0.1          110      00:09:22
    192.168.0.2          110      00:04:08
    202.1.99.1           110      00:04:08
    201.1.40.1           110      00:04:08
  Distance: (default is 110)
```

Pentru celelalte rutere, Router ID-urile alese automat sunt următoarele:

Mercury: Router ID 172.30.0.1

Venus: Router ID 201.1.40.1

Earth: Router ID 202.1.99.1

Mars: Router ID 192.168.0.1

Jupiter: Router ID 192.168.0.2

Schimbarea de update-uri de rutare între rutere vecine OSPF se face doar după stabilirea unei adiacențe. Comanda `show ip ospf neighbors` din exemplul următor listează starea adiacențelor ruterului Venus pe interfețele sale participante in OSPF:

```
Venus#show ip ospf neighbors
Neighbor ID     Pri   State           Dead Time   Address         Interface
172.30.0.1        1   FULL/DR         00:00:35    10.0.0.3  FastEthernet0/0
202.1.99.1        1   FULL/BDR        00:00:31    10.0.0.1  FastEthernet0/0
172.30.0.1        0   FULL/  -        00:00:33    172.30.0.1      Serial1/0
```

Pentru Venus, Mercur este un vecin conectat printr-o legătură point-to-point, pentru care stabilirea unui DR/BDR nu este necesară. Pentru rețeaua switched dintre Venus, Mercur și Earth, fiind un mediu multiacces, se alege un DR și un BDR care vor reprezenta această rețea prin LSA-urile lor. Din output-ul de mai sus se poate observa că Venus a stabilit o adiacență de tip DR cu vecinul cu ID-ul 172.30.0.1 (adică Mercury, conform listei de mai sus) una de tip BDR cu Earth. Din acest output se deduce faptul că Mercury a fost ales DR iar Earth a fost ales BDR al segmentului multiacces respectiv.

Din perspectiva lui Mercury, și acesta a stabilit două tipuri de relații de adiacență cu Venus, prin legătura serială și cea multiacces. În segmentul de rețea dintre cele trei rutere, Venus se află în relația DROTHER cu Mercury deoarece nu este nici DR și nici BDR.

```
Mercury#show ip ospf neighbors

Neighbor ID   Pri   State           Dead Time   Address         Interface
201.1.40.1      0   FULL/  -        00:00:38    172.30.0.2      Serial1/0
201.1.40.1      1   FULL/DROTHER    00:00:39    10.0.0.2  FastEthernet0/0
202.1.99.1      1   FULL/BDR        00:00:35    10.0.0.1  FastEthernet0/0
```

Parametrii OSPF configurați pe fiecare interfață, precum și situația DR/BDR pe segmentul respectiv, pot fi vizualizate prin comanda `show ip ospf interface`. Exemplificare pentru Mercury:

```
Venus#show ip ospf interface fastethernet0/0
FastEthernet0/0 is up, line protocol is up
  Internet Address 10.0.0.2/24, Area 0
  Process ID 1, Router ID 201.1.40.1, Network Type BROADCAST, Cost: 1
  Transmit Delay is 1 sec, State DROTHER, Priority 1
  Designated Router (ID) 172.30.0.1, Interface address 10.0.0.3
  Backup Designated router (ID) 202.1.99.1, Interface address 10.0.0.1
  Timer intervals configured, Hello 10, Dead 40, Wait 40, Retransmit 5
    oob-resync timeout 40
    Hello due in 00:00:07
  Index 2/2, flood queue length 0
  Next 0x0(0)/0x0(0)
```

```
  Last flood scan length is 0, maximum is 1
  Last flood scan time is 0 msec, maximum is 0 msec
  Neighbor Count is 2, Adjacent neighbor count is 2
    Adjacent with neighbor 172.30.0.1  (Designated Router)
    Adjacent with neighbor 202.1.99.1  (Backup Designated Router)
  Suppress hello for 0 neighbor(s)
```

Alegerea DR-ului și a BDR-ului pe segmentul de rețea multiacces în cauză se face, în principiu, pe baza priorității configurate pe fiecare interfață. Din moment ce prioritățile celor 3 rutere sunt la valoarea implicită (1), decizia se ia pe baza Router ID-ului. După cum se poate observa, și de prioritățile se află la valorile implicite, ruterul ales ca DR nu este ruterul cu Router ID-ul cel mai mare. Acest lucru se datorează faptului că cele 3 rutere nu au pornit simultan, astfel că primul ruter s-a declarat DR. Alegerea DR-ului în OSPF nu e preemptivă, deci un ruter cu un Router ID superior sau o prioritate mai bună nu va deveni DR atâta timp cât nu este primul ruter care pornește pe segmentul respectiv.

Comanda pentru modificarea priorității unui ruter este `ip ospf priority`, se introduce explicit pe interfața conectată la segmentul multiacces și acceptă valori între 0 și 255. De asemenea, pentru alterarea manuală a deciziei de alegere a DR/BDR pe baza Router ID-urilor, acestea pot fi modificate prin comanda următoare:

```
Mercury(config)#router ospf 1
Mercury(config-router)#router-id 172.30.0.1
```

În momentul de față, ruterul Mercury este DR iar ruterul Earth este BDR. Pentru a putea forța alegerea ruterului Earth ca DR pentru segmentul Ethernet se va realiza resetarea procesului de OSPF pe ruterul Mercury, ceea ce va avea ca efect și resetarea adiacențelor cu celelalte rutere.

```
Mercury#clear ip ospf process
Reset ALL OSPF processes? [no]: y
```

Mesajele de syslog indică adiacențele căzute:

```
Mercury#
*Mar  1 03:00:40.551: %OSPF-5-ADJCHG: Process 1, Nbr 201.1.40.1 on
Serial1/0 from FULL to DOWN, Neighbor Down: Interface down or detached
*Mar  1 03:00:40.555: %OSPF-5-ADJCHG: Process 1, Nbr 201.1.40.1 on
FastEthernet0/0 from 2WAY to DOWN, Neighbor Down: Interface down or
detached
*Mar  1 03:00:40.555: %OSPF-5-ADJCHG: Process 1, Nbr 202.1.99.1 on
FastEthernet0/0 from FULL to DOWN, Neighbor Down: Interface down or
detached
```

După câteva secunde, adicențele revin pentru fiecare vecin în parte:

```
*Mar  1 03:00:40.607: %OSPF-5-ADJCHG: Process 1, Nbr 201.1.40.1 on
Serial1/0 from LOADING to FULL, Loading Done
*Mar  1 03:00:59.567: %OSPF-5-ADJCHG: Process 1, Nbr 201.1.40.1 on
FastEthernet0/0 from LOADING to FULL, Loading Done
*Mar  1 03:01:00.611: %OSPF-5-ADJCHG: Process 1, Nbr 202.1.99.1 on
FastEthernet0/0 from LOADING to FULL, Loading Done
```

Analizând rezultatul de pe interfața lui Venus, obținem următoarele informații:

```
Venus#sh ip ospf inte fa0/0 | inc Adj
 Neighbor Count is 2, Adjacent neighbor count is 2
    Adjacent with neighbor 172.30.0.1
    Adjacent with neighbor 202.1.99.1  (Designated Router)
```

Ruterul Earth a fost BDR și a devenit automat DR după ce a pierdut adiacența cu DR -ul precedent.

Identificarea ruterelor dintr-o topologie OSPF după Router ID poate fi dificilă. OSPF pune la dispoziție o metodă de rezolvare a adreselor IP din output-urile comenzilor sale astfel încât în locul acestora să fie afișate, spre exemplu, hostname-urile ruterelor vecine.

Spre exemplu, pentru ruterul Earth se folosește următoarea comandă:

```
Earth(config)#ip ospf name-lookup
```

În continuare, se definesc intrări statice pentru toate Router ID-urile ruterelor vecine:

```
Earth(config)#ip host Mercury 172.30.0.1
Earth(config)#ip host Venus 201.1.40.1
Earth(config)#ip host Mars 192.168.0.1
```

Putem testa rezultatul prin comanda următoare:

```
Earth(config)#do show ip ospf neighbors

Neighbor ID     Pri   State           Dead Time   Address
Interface
Mars              0   FULL/  -        00:00:35    172.20.0.2
Serial1/0
Mercury           1   FULL/DROTHER    00:00:37    10.0.0.3
FastEthernet0/0
Venus             1   FULL/BDR        00:00:34    10.0.0.2
FastEthernet0/0
```

Pentru topologia de mai sus sunt transmise doar două tipuri de LSA-uri, de tip 1 și 2. Se poate verifica acest lucru de pe orice ruter:

```
Earth#sh ip ospf database

            OSPF Router with ID (202.1.99.1) (Process ID 1)

            Router Link States (Area 0)

Link ID         ADV Router      Age         Seq#       Checksum Link count
172.30.0.1      172.30.0.1      1430        0x80000012 0x0064E7 3
192.168.0.1     192.168.0.1     237         0x8000000A 0x00B877 3
192.168.0.2     192.168.0.2     160         0x80000008 0x00EA93 1
201.1.40.1      201.1.40.1      1450        0x80000012 0x0062B8 12
202.1.99.1      202.1.99.1      1449        0x80000011 0x003FE5 7

            Net Link States (Area 0)

Link ID         ADV Router      Age         Seq#       Checksum
10.0.0.1        202.1.99.1      1429        0x80000002 0x00D039
192.168.0.2     192.168.0.2     160         0x80000007 0x0092F4
```

Se observă faptul că LSA-urile de tip 2 sunt transmise de către DR-urile fiecărei rețele multiacces (rețeaua 10.0.0.0/24 cu ruterul Earth, respectiv rețeaua 192 .168.0.0/24 cu ruterul Jupiter).

5.4.2.2 Rute default, redistribuire

Pentru a configura ruterul Mercury să injecteze o rută default în topologie se folosește comanda următoare:

```
Mercury(config-router)#default-information originate
```

Verificând oricare tabelă de rutare, chiar și a vecinilor, se observă că ruta default nu a fost propagată:

```
Venus#show ip route | include Gateway
Gateway of last resort is not set
```

Mercury nu va propaga o rută default atâta timp cât el însuși nu deține o rută default în tabela sa de rutare. Pentru a nu ține cont de această condiție, comanda poate fi introdusă în felul următor:

```
Mercury(config-router)#default-information originate always

Venus#show ip route
[…]
Gateway of last resort is 10.0.0.3 to network 0.0.0.0
[…]
O*E2 0.0.0.0/0 [110/1] via 10.0.0.3, 00:00:03, FastEthernet0/0
```

Evident, se poate defini și o rută default proprie -zisă pe ruterul Mercury, iar `default-information originate` va propaga acea rută.

Pentru a analiza modul de redistribuire a rutelor, se va crea o rută statică spre o destinație fictivă și se va redistribui în procesul de OSPF:

```
Mercury(config)#ip route 99.99.99.99 255.255.255.255 null0
Mercury(config)#router ospf 1
Mercury(config-router)#redistribute static subnets
```

Parametrul `subnets` este obligatoriu pentru a evita redistribuirea exclusivă a rutelor classful.

Rutele externe vor apărea în mod implicit în tabela de rutare ca fiind rute de tip O E2. Și în exemplul anterior, în care o rută default a fost injectată în topologie, redistribuirea este echivalentă cu introducerea în OSPF a unor rute din exteriorul domeniului său de rutare. Pentru confirmare, se poate verifica tabela de rutare de pe Venus, care indică LSA-urile de tip 5 corespunzătoare rutei default și celei redistribuite:

```
Venus#show ip ospf database

            OSPF Router with ID (201.1.40.1) (Process ID 1)

            Router Link States (Area 0)

Link ID         ADV Router      Age         Seq#       Checksum Link count
172.30.0.1      172.30.0.1      1010        0x80000016 0x0062E3 3
192.168.0.1     192.168.0.1     1725        0x8000000C 0x00B479 3
192.168.0.2     192.168.0.2     1672        0x8000000A 0x00E695 1
201.1.40.1      201.1.40.1      1133        0x80000015 0x005CBB 12
202.1.99.1      202.1.99.1      1025        0x80000014 0x0039E8 7

            Net Link States (Area 0)

Link ID         ADV Router      Age         Seq#       Checksum
10.0.0.1        202.1.99.1      1025        0x80000005 0x00CA3C
192.168.0.2     192.168.0.2     1672        0x80000009 0x008EF6

            Type-5 AS External Link States

Link ID         ADV Router      Age         Seq#       Checksum Tag
0.0.0.0         172.30.0.1      487         0x80000002 0x0030B5 1
99.99.99.99     172.30.0.1      254         0x80000001 0x000344 0
```

Metrica rutei redistribuite, din perspectiva lui Venus, spre exemplu, este 20. Verificând aceeași rută și pe Jupiter, se constată că aceasta continuă să aibă aceeași valoare, constantă, indiferent de ruterul care o introduce în tabela de rutare:

```
Venus#show ip route ospf
[...]
99.0.0.0/32 is subnetted, 1 subnets
O E2    99.99.99.99 [110/20] via 10.0.0.3, 00:01:05, FastEthernet0/0
```

```
Jupiter#show ip route ospf
[...]
99.0.0.0/32 is subnetted, 1 subnets
O E2    99.99.99.99 [110/20] via 192.168.0.1, 00:01:55, FastEthernet0/0
```

Metrica implicită, de 20, a rutelor externe poate fi modificată atât global cât și pentru rutele explicit redistribuite. În cazul de față, următoarele două comenzi introduse pe Mercury au același rezultat:

```
Mercury(config-router)#default-metric 99
Mercury(config-router)#redistribute static subnets metric 99

Jupiter#show ip route ospf
[...]
99.0.0.0/32 is subnetted, 1 subnets
O E2    99.99.99.99 [110/99] via 192.168.0.1, 00:01:55, FastEthernet0/0
```

O valoare constantă a metricii este suficientă în momentul în care există un singur ASBR în topologie, deoarece nu există o alternativă pentru ieșirea spre rute externe. Dacă, însă, mai multe ASBR-uri introduc rute externe către aceleași destinații, ruterele trebuie să poată alege cea mai eficientă cale spre o anumită destinație externă. Pentru aceasta, rutele redistribuite pot fi convertite la tipul E1, care adună la metrica implicită și suma metricilor din interiorul domeniului OSPF[1]:

```
Mercury(config-router)#redistribute static subnets metric 99 metric-type 1
```

Comanda introdusă ca și anterior, cu parametrul `metric-type 1`, va introduce ruta externă ca o rută E1. Rezultatul poate fi confirmat pe Venus și Jupiter:

```
Venus#show ip route ospf
[...]
99.0.0.0/32 is subnetted, 1 subnets
O E1    99.99.99.99 [110/100] via 10.0.0.3, 00:00:06, FastEthernet0/0

Jupiter#show ip route ospf
[...]
99.0.0.0/32 is subnetted, 1 subnets
O E1    99.99.99.99 [110/165] via 192.168.0.1, 00:00:15, FastEthernet0/0
```

Pentru Venus, ruta are valoarea 100 deoarece acesta adaugă 1 ($10^8/10^8$ pentru rețele FastEthernet) la metrica 99 de redistribuire. Jupiter vede rețeaua cu metrica 165, obținută din 99 + 2x1 + 64, adică două segmente FastEthernet și un segment de linie serială ($10^8/(1544*10^3)$ = 64 Kbps pentru o linie serială de viteză T1).

5.4.3 Configurare OSPF Multi-area

Pentru a converti topologia anterioară la una Multi-area, se vor face următoarele modificări:

Interfețele loopback definite pe ruterul Venus vor fi trecute în Area 100;

Interfețele loopback definite pe ruterul Earth vor fi trecute în Area 200;

Rețeaua multiacces dintre Mercury, Venus și Earth rămâne în Area 0, împreună cu legătura dintre Mercury și Venus;

Legătura dintre Earth și Mars devine Area 7;

Rețeaua dintre Mars și Jupiter devine extensie pentru Area 7.

---

[1] Se observă faptul că aceeași comandă este folosită cu același efect și de către IS-IS.

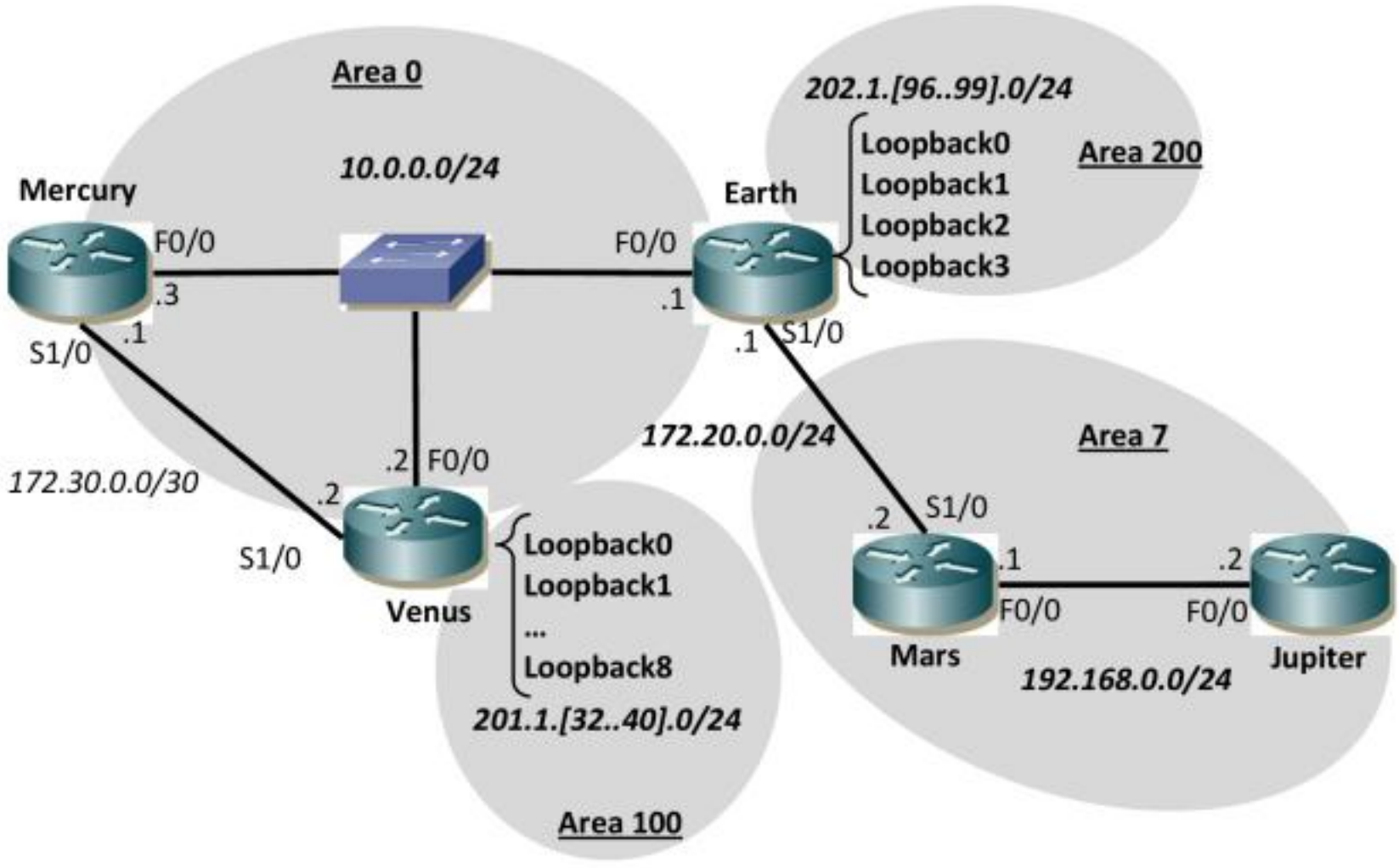

Figura 5-28: Topologie multi-area

Configurația de bază pentru OSPF Multi-area, pe cele 5 rutere, este următoarea:

```
Mercury#sh run | b router
router ospf 1
 router-id 172.30.0.1
redistribute static metric 99 metric-type 1 subnets
 network 10.0.0.0 0.0.0.255 area 0
 network 172.30.0.0 0.0.0.3 area 0
 default-information originate always
 default-metric 99

Venus#sh run | b router
router ospf 1
network 10.0.0.0 0.0.0.255 area 0
 network 172.30.0.0 0.0.0.3 area 0
 network 201.1.0.0 0.0.255.255 area 100

Earth#sh run | b router
router ospf 1
network 10.0.0.0 0.0.0.255 area 0
 network 172.20.0.0 0.0.0.3 area 7
 network 202.1.0.0 0.0.255.255 area 200

Mars#sh run | b router
router ospf 1
network 172.20.0.0 0.0.0.3 area 7
 network 192.168.0.0 0.0.0.255 area 7

Jupiter#sh run | b router
router ospf 1
network 192.168.0.0 0.0.0.255 area 7
```

Într-o topologie Multi-area, rețelele din arii diferite sunt anunțate prin LSA-uri de tip 3 (summary). Pentru ruterul Jupiter din area 7, spre exemplu, rețelele din ariile 0, 100 și 200 sunt primite ca summary LSA, după cum se observă și din link state database-ul său:

```
Jupiter#sh ip ospf data
```

```
          OSPF Router with ID (192.168.0.2) (Process ID 1)

          Router Link States (Area 7)
Link ID         ADV Router      Age         Seq#       Checksum Link count
192.168.0.1     192.168.0.1     27          0x80000005 0x00C272 3
192.168.0.2     192.168.0.2     58          0x80000004 0x00F28F 1
202.1.99.1      202.1.99.1      337         0x80000004 0x00C78E 2

          Net Link States (Area 7)
Link ID         ADV Router      Age         Seq#       Checksum
192.168.0.2     192.168.0.2     58          0x80000003 0x009AF0

          Summary Net Link States (Area 7)
Link ID         ADV Router      Age         Seq#       Checksum
10.0.0.0        202.1.99.1      337         0x80000003 0x0026D9
172.30.0.0      202.1.99.1      337         0x80000003 0x00EA17
201.1.32.1      202.1.99.1      337         0x80000003 0x00FB21
201.1.33.1      202.1.99.1      337         0x80000003 0x00F02B
201.1.34.1      202.1.99.1      337         0x80000003 0x00E535
201.1.35.1      202.1.99.1      337         0x80000003 0x00DA3F
201.1.36.1      202.1.99.1      340         0x80000003 0x00CF49
201.1.37.1      202.1.99.1      340         0x80000003 0x00C453
201.1.38.1      202.1.99.1      340         0x80000003 0x00B95D
201.1.39.1      202.1.99.1      340         0x80000003 0x00AE67
201.1.40.1      202.1.99.1      340         0x80000003 0x00A371
202.1.96.1      202.1.99.1      1580        0x80000004 0x0020BB
202.1.97.1      202.1.99.1      1580        0x80000004 0x0015C5
202.1.98.1      202.1.99.1      1580        0x80000004 0x000ACF
202.1.99.1      202.1.99.1      1580        0x80000004 0x00FED9

          Summary ASB Link States (Area 7)
Link ID         ADV Router      Age         Seq#       Checksum
172.30.0.1      202.1.99.1      340         0x80000003 0x0062DA

          Type-5 AS External Link States
Link ID         ADV Router      Age         Seq#       Checksum Tag
0.0.0.0         172.30.0.1      676         0x8000000E 0x0018C1 1
99.99.99.99     172.30.0.1      928         0x8000000A 0x0086E8 0
```

În continuare, rutele redistribuite sunt primite prin LSA-uri de tip 5.

De asemenea, LSA-urile de tip 3 creează rute de tip inter-area (IA) vizibile și în tabela de rutare:

```
Jupiter#sh ip route
[...]
     202.1.96.0/32 is subnetted, 1 subnets
O IA    202.1.96.1 [110/66] via 192.168.0.1, 01:09:56, FastEthernet0/0
     201.1.39.0/32 is subnetted, 1 subnets
O IA    201.1.39.1 [110/67] via 192.168.0.1, 01:18:10, FastEthernet0/0
     202.1.97.0/32 is subnetted, 1 subnets
O IA    202.1.97.1 [110/66] via 192.168.0.1, 01:09:56, FastEthernet0/0
     201.1.38.0/32 is subnetted, 1 subnets
O IA    201.1.38.1 [110/67] via 192.168.0.1, 01:18:10, FastEthernet0/0
     202.1.98.0/32 is subnetted, 1 subnets
O IA    202.1.98.1 [110/66] via 192.168.0.1, 01:09:56, FastEthernet0/0
     201.1.37.0/32 is subnetted, 1 subnets
[...]
```

5.4.3.1 Sumarizarea rutelor

Ruterele Venus și Earth sunt ABR-uri ce conectează ariile 100, respectiv 200 la backbone (Area 0). ABR-urile pot realiza sumarizare inter-area. În cazul de față, este convenabilă sumarizarea rețelelor de loopback de pe ambele rutere într-o singură rețea generică, urmând ca doar această rețea să fie anunțată în Area 0.

```
Venus#show ip interface brief | inc Loop
Loopback0              201.1.32.1      YES NVRAM  up                    up
```

```
Loopback1              201.1.33.1      YES NVRAM  up                    up
Loopback2              201.1.34.1      YES NVRAM  up                    up
Loopback3              201.1.35.1      YES NVRAM  up                    up
Loopback4              201.1.36.1      YES NVRAM  up                    up
Loopback5              201.1.37.1      YES NVRAM  up                    up
Loopback6              201.1.38.1      YES NVRAM  up                    up
Loopback7              201.1.39.1      YES NVRAM  up                    up
Loopback8              201.1.40.1      YES NVRAM  up                    up
Venus#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Venus(config)#router ospf 1
Venus(config-router)#area 100 range 201.1.32.0 255.255.240.0
```

Pe Venus, rețelele interfețelor looback au fost sumarizate la adresa 201.1.32.0/20. Acest prefix este acum vizibil atât în Area 0 cât și în celelalte arii. O sumariza re similară se realizează și pentru rețelele ruterului Earth:

```
Earth#show ip int brief | inc Loop
Interface      IP-Address      OK? Method Status                Protocol
Loopback0      202.1.96.1      YES NVRAM  up                    up
Loopback1      202.1.97.1      YES NVRAM  up                    up
Loopback2      202.1.98.1      YES NVRAM  up                    up
Loopback3      202.1.99.1      YES NVRAM  up                    up
Earth#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
Earth(config)#router ospf 1
Earth(config-router)#area 200 range 202.1.96.0 255.255.252.0
```

Earth anunță în Area 0 rețeaua sumarizată 202.1.96.0/22. În final, rutele sumarizate sunt anunțate în toate celelalte arii:

```
Jupiter#sh ip route
[...]
     99.0.0.0/32 is subnetted, 1 subnets
O E1    99.99.99.99 [110/165] via 192.168.0.1, 00:01:22, FastEthernet0/0
     172.20.0.0/30 is subnetted, 1 subnets
O       172.20.0.0 [110/65] via 192.168.0.1, 01:30:54, FastEthernet0/0
     172.30.0.0/30 is subnetted, 1 subnets
O IA    172.30.0.0 [110/130] via 192.168.0.1, 01:30:54, FastEthernet0/0
     10.0.0.0/24 is subnetted, 1 subnets
O IA    10.0.0.0 [110/66] via 192.168.0.1, 01:30:54, FastEthernet0/0
C    192.168.0.0/24 is directly connected, FastEthernet0/0
O*E2 0.0.0.0/0 [110/1] via 192.168.0.1, 00:01:22, FastEthernet0/0
O IA 202.1.96.0/22 [110/66] via 192.168.0.1, 00:01:27, FastEthernet0/0
O IA 201.1.32.0/20 [110/67] via 192.168.0.1, 00:07:58, FastEthernet0/0
```

5.4.3.2 Arii Stub

Presupunând că ruterele din aria 7 sunt mai puțin performante, se poate lua decizia de a converti această arie într-un Stub. Se specifică acest lucru pe toate ruterele care au cel puțin o interfață în aria respectivă. Drept rezultat, aria 7 nu va mai primi rutele anunțate prin LSA-uri de tip 5 (rutele externe, redistribuite) și va primi doar o rută default în locul lor. Configurarea se face prin comenzile următoare:

```
Earth(config-router)#area 7 stub
Mars(config-router)#area 7 stub
Jupiter(config-router)#area 7 stub
```

Se observă că, în tabela de rutare a lui Jupiter, cele două rute externe au fost înlocuite de o rută default, inter-area:

```
Jupiter(config-router)#do sh ip route
Gateway of last resort is 192.168.0.1 to network 0.0.0.0
     172.20.0.0/30 is subnetted, 1 subnets
```

```
O       172.20.0.0 [110/65] via 192.168.0.1, 00:00:12, FastEthernet0/0
     172.30.0.0/30 is subnetted, 1 subnets
O IA    172.30.0.0 [110/130] via 192.168.0.1, 00:00:12, FastEthernet0/0
     10.0.0.0/24 is subnetted, 1 subnets
O IA    10.0.0.0 [110/66] via 192.168.0.1, 00:00:12, FastEthernet0/0
C    192.168.0.0/24 is directly connected, FastEthernet0/0
O*IA 0.0.0.0/0 [110/66] via 192.168.0.1, 00:00:12, FastEthernet0/0
O IA 202.1.96.0/22 [110/66] via 192.168.0.1, 00:00:12, FastEthernet0/0
O IA 201.1.32.0/20 [110/67] via 192.168.0.1, 00:00:12, FastEthernet0/0
```

Mai mult chiar, ruterele din aria 7 pot primi o rută default pentru toate rutele din afara ariei 7, nu doar cele externe. Pentru aceasta, se specifică pe ABR (Earth) faptul că această arie nu va accepta LSA-uri de tip 3, creând astfel o arie Totally Stubby:

```
Earth(config-router)#area 7 stub no-summary
```

După cum se observă și din sintaxa comenzii, ABR -ul nu va mai injecta în aria 7 rutele de tip summary (tip 3) cu excepția unei singure rute, cea default. Comanda poate fi da tă doar pe ABR deoarece acesta este responsabil cu introducerea LSA-urilor de tip 3.

Tabela de rutare de pe Jupiter arată acum doar rutele direct conectate, cele locale (din Area 7), precum și ruta default ce indică direcția spre ABR:

```
Jupiter(config-router)#do sh ip route

Gateway of last resort is 192.168.0.1 to network 0.0.0.0

     172.20.0.0/30 is subnetted, 1 subnets
O       172.20.0.0 [110/65] via 192.168.0.1, 00:06:50, FastEthernet0/0
C    192.168.0.0/24 is directly connected, FastEthernet0/0
O*IA 0.0.0.0/0 [110/66] via 192.168.0.1, 00:00:05, FastEthernet0/0
```

De asemenea, în link state database LSA-ul rutei default este singurul LSA de tip 3:

```
Jupiter#sh ip ospf data

            OSPF Router with ID (192.168.0.2) (Process ID 1)

            Router Link States (Area 7)
Link ID         ADV Router      Age         Seq#       Checksum Link count
192.168.0.1     192.168.0.1     674         0x80000008 0x00DA59 3
192.168.0.2     192.168.0.2     389         0x80000007 0x000B76 1
202.1.99.1      202.1.99.1      682         0x80000007 0x00DF75 2

            Net Link States (Area 7)
Link ID         ADV Router      Age         Seq#       Checksum
192.168.0.2     192.168.0.2     389         0x80000006 0x00B2D7

            Summary Net Link States (Area 7)
Link ID         ADV Router      Age         Seq#       Checksum
0.0.0.0         202.1.99.1      264         0x80000002 0x00C844
```

Pentru a putea păstra caracteristica de Stub a ariei 7, dar a permite injectarea de rute externe, aceasta poate fi configurată ca o arie Not-So-Stubby (NSSA). O arie NSSA ce nu acceptă decât o rută default pentru rutele externeși cele din alte arii este o arie NSSA Totally Stubby. Singurele LSA -uri acceptate într-o astfel de arie sunt cele de tip 1, 2, ruta default ca singurul LSA de tip 3 și LSA -urile de tip 7 reprezentând rutele externe injectate prin aria NSSA.

```
Earth(config-router)#no area 7 stub
Earth(config-router)#area 7 nssa no-summary
Mars(config-router)#no area 7 stub
Mars(config-router)#area 7 nssa
Jupiter(config-router)#no area 7 stub
Jupiter(config-router)#area 7 nssa
```

Pentru a demonstra prezența LSA-urilor de tip 7 se redistribuie o rută statică de pe Jupiter:

```
Jupiter(config)#ip route 133.33.33.0 255.255.255.0 null0
Jupiter(config)#router ospf 1
Jupiter(config-router)#redistribute static subnets metric 180
```

Tabela de rutare de pe Mars arată această rută anunțată printr -un LSA de tip 7 ca fiind de tip N2:

```
Mars(config-router)#do sh ip route
Gateway of last resort is 172.20.0.1 to network 0.0.0.0
     172.20.0.0/30 is subnetted, 1 subnets
C       172.20.0.0 is directly connected, Serial1/0
C    192.168.0.0/24 is directly connected, FastEthernet0/0
     133.33.0.0/24 is subnetted, 1 subnets
O N2    133.33.33.0 [110/180] via 192.168.0.2, 00:10:01, FastEthernet0/0
O*IA 0.0.0.0/0 [110/65] via 172.20.0.1, 00:10:01, Serial1/0
```

De asemenea, în baza sa de date link-state, ruta este identificată printr-un LSA de tip 7:

```
Mars(config-router)#do sh ip ospf data
            OSPF Router with ID (192.168.0.1) (Process ID 1)

            Router Link States (Area 7)

Link ID         ADV Router      Age         Seq#       Checksum Link count
192.168.0.1     192.168.0.1     676         0x8000000B 0x005CCC 3
192.168.0.2     192.168.0.2     676         0x80000009 0x0094E0 1
202.1.99.1      202.1.99.1      685         0x8000000A 0x0067E0 2

            Net Link States (Area 7)

Link ID         ADV Router      Age         Seq#       Checksum
192.168.0.2     192.168.0.2     677         0x80000008 0x00364A

            Summary Net Link States (Area 7)

Link ID         ADV Router      Age         Seq#       Checksum
0.0.0.0         202.1.99.1      711         0x80000004 0x004CB6

            Type-7 AS External Link States (Area 7)

Link ID         ADV Router      Age         Seq#       Checksum Tag
133.33.33.0     192.168.0.2     682         0x80000001 0x000C4B 0
```

Deoarece LSA-urile de tip 7 nu sunt permise decât în ariile de tip NSSA, la părăsirea acestora ABR-urile le convertesc în LSA-uri de tip 5. În consecință, Mercury, spre exemplu, va primi ruta spre 133.33.33.0/24 cu tipul E2:

```
Mercury#sh ip route
[...]
O E2    133.33.33.0 [110/180] via 10.0.0.1, 00:12:25, FastEthernet0/0
O IA 202.1.96.0/22 [110/2] via 10.0.0.1, 00:13:09, FastEthernet0/0
O IA 201.1.32.0/20 [110/2] via 10.0.0.2, 00:13:09, FastEthernet0/0
```

Într-adevăr, ruta este anunțată printr-un LSA de tip 5, conform bazei de date de pe Mercury:

```
Mercury#sh ip ospf database | begin Type-5
            Type-5 AS External Link States

Link ID         ADV Router      Age         Seq#       Checksum Tag
0.0.0.0         172.30.0.1      403         0x80000010 0x0014C3 1
99.99.99.99     172.30.0.1      655         0x8000000C 0x0082EA 0
133.33.33.0     202.1.99.1      880         0x80000001 0x00D1CA 0
```

5.4.3.3 Autentificare

Pentru demonstrarea capabilităților de autentificare OSPF, pachetele protocolului de rutare vor fi autentificate după următoarea schemă:

Area 0: autentificare MD5; ruterele vor calcula un hash pe baza parolei și a conținutului fiecărui pachet iar parola nu va fi transmisă clear-text. De asemenea, pachetele vor fi protejate împotriva atacurilor de tip replay prin numere de secvență.

Area 7: autentificare clear-text; parola va fi transmisă în clar împreună cu pachetele.

Exemplificare pentru Area 0 și 7: ruterele Mercury, Earth și Mars:

Mercury:

```
Mercury#show run
interface FastEthernet0/0
 ip address 10.0.0.3 255.255.255.0
 ip ospf message-digest-key 1 md5 cisco
!
interface Serial1/0
 ip address 172.30.0.1 255.255.255.252
 ip ospf message-digest-key 1 md5 cisco
!
router ospf 1
 router-id 172.30.0.1
area 0 authentication message-digest
 redistribute static metric 99 metric-type 1 subnets
 network 10.0.0.0 0.0.0.255 area 0
 network 172.30.0.0 0.0.0.3 area 0
 default-information originate always
 default-metric 99
```

Earth:

```
Earth#show run
interface FastEthernet0/0
 ip address 10.0.0.1 255.255.255.0
 ip ospf message-digest-key 1 md5 cisco
!
interface Serial1/0
 ip address 172.20.0.1 255.255.255.252
 ip ospf authentication
 ip ospf authentication-key ciscokey
!
router ospf 1
area 0 authentication message-digest
 area 7 authentication
 area 7 nssa no-summary
 area 200 range 202.1.96.0 255.255.252.0
 network 10.0.0.0 0.0.0.255 area 0
 network 172.20.0.0 0.0.0.3 area 7
 network 202.1.0.0 0.0.255.255 area 200
```

Mars:

```
Mars#show run
interface FastEthernet0/0
 ip address 192.168.0.1 255.255.255.0
 ip ospf authentication
 ip ospf authentication-key ciscolan
!
interface Serial1/0
 ip address 172.20.0.2 255.255.255.252
 ip ospf authentication
 ip ospf authentication-key ciscokey
!
router ospf 1
area 7 authentication
```

```
area 7 nssa
network 172.20.0.0 0.0.0.3 area 7
network 192.168.0.0 0.0.0.255 area 7
```

Indiferent de metoda de autentificare folosită, cheia trebuie să fie aceeași doar între vecinii din același segment, după cum se observă din configurația lui Mars, care are configurate cheile `ciscolan` și `ciscokey` pe interfețe diferite aflate în aceeași arie.

5.4.3.4 Legături virtuale

Configurarea unei legături virtuale este utilă în una dintre situațiile următoare:

Imposibilitatea fizică a conectării unei arii la aria de backbone (Area 0);

Conectarea a două arii de backbone separate.

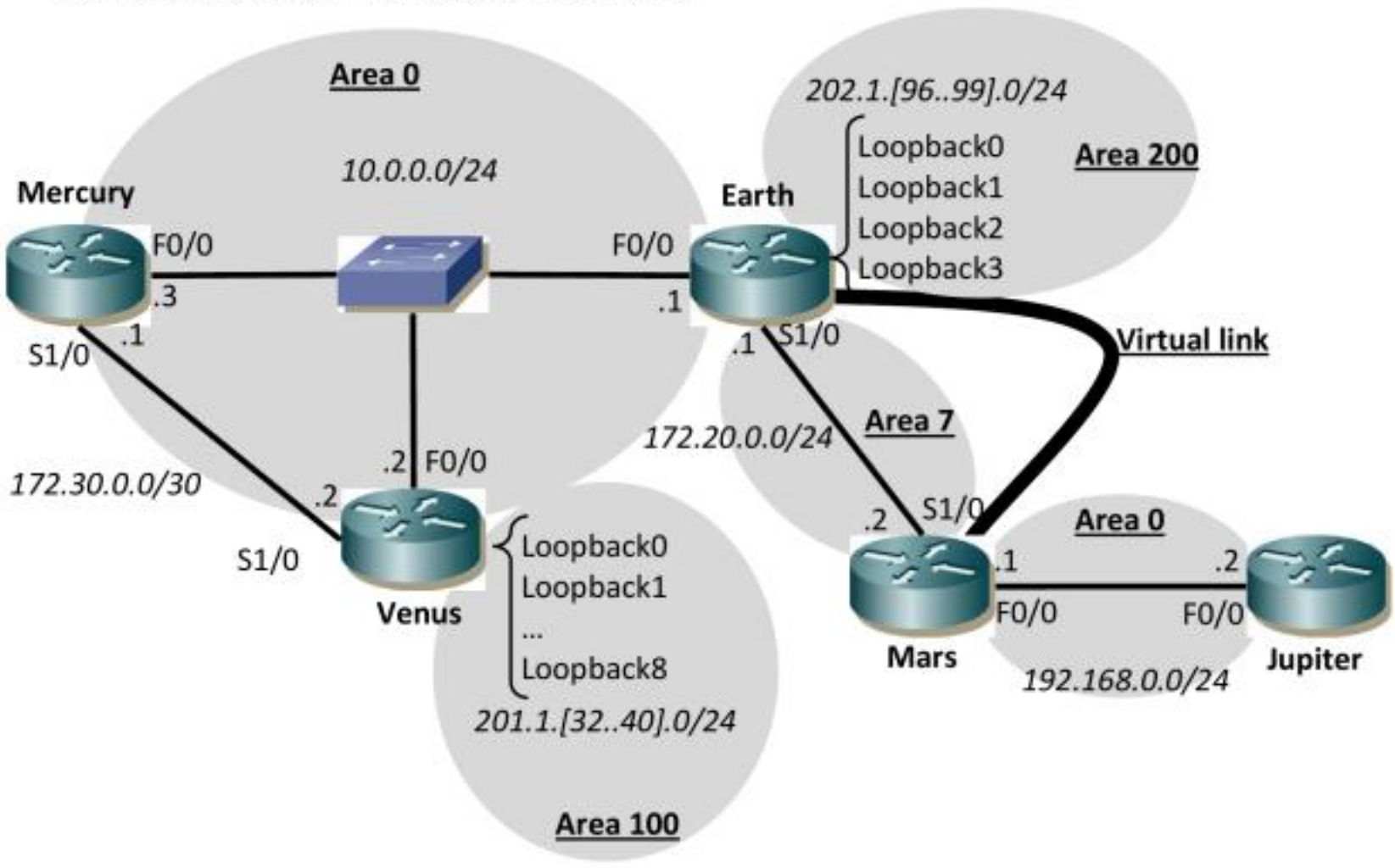

Figura 5-29: Topologie multi-area cu virtual link

Pentru topologia curentă, legătura dintre Marși Jupiter va fi trecută în Area 0 iar legătura dintre Earth și Mars va rămâne în Area 7. Cele două arii 0 distincte trebuie conectate cu o legătură virtuală configurată între Earthși Mars. Totodată, trebuie avute în vedere două elemente de configurație ce trebuie rezolvate înainte ca legătura să fie funcțională:

Area 7 nu poate fi arie de tranzit pentru o legătură virtuală atâta timp cât este o arie Stub (de orice fel), deoarece ariile stub nu pot conține rute externe și, deci, nu pot fi folosite corect drept arii de tranzit.

După configurarea segmentului dintre Marși Jupiter ca membru în Area 0, interfețele din acest segment trebuie configurate cu aceiași parametri de autentificare ca și Area 0. De asemenea, legătura virtuală trebuie sa specifice faptul că vor fi transportate pachete autentificate.

În continuare sunt exemplificate configurările relevante pentru crearea legăturii virtuale pentru ruterele Earth, Mars și Jupiter:

Earth:

```
Earth#show run
interface Serial1/0
 ip address 172.20.0.1 255.255.255.252
 ip ospf authentication
 ip ospf authentication-key ciscokey
!
router ospf 1
area 0 authentication message-digest
 area 7 authentication
 area 7 virtual-link 192.168.0.1 message-digest-key 1 md5 cisco
 area 200 range 202.1.96.0 255.255.252.0
 network 10.0.0.0 0.0.0.255 area 0
 network 172.20.0.0 0.0.0.3 area 7
 network 202.1.0.0 0.0.255.255 area 200
```

Mars:

```
Mars#show run
interface FastEthernet0/0
 ip address 192.168.0.1 255.255.255.0
 ip ospf authentication
 ip ospf authentication-key ciscolan
 ip ospf message-digest-key 1 md5 cisco
!
interface Serial1/0
 ip address 172.20.0.2 255.255.255.252
 ip ospf authentication
 ip ospf authentication-key ciscokey
!
router ospf 1
area 0 authentication message-digest
 area 1 virtual-link 202.1.99.1
 area 7 authentication
 area 7 virtual-link 202.1.99.1 message-digest-key 1 md5 cisco
 network 172.20.0.0 0.0.0.3 area 7
 network 192.168.0.0 0.0.0.255 area 0
```

Jupiter:

```
Jupiter#show run
interface FastEthernet0/0
 ip address 192.168.0.2 255.255.255.0
 ip ospf authentication
 ip ospf authentication-key ciscolan
 ip ospf message-digest-key 1 md5 cisco
!
router ospf 1
area 0 authentication message-digest
 area 7 authentication
 area 7 nssa
 redistribute static metric 180 subnets
 network 192.168.0.0 0.0.0.255 area 0
```

Configurarea parțială sau lipsa configurării unei legături virtuale în condițiile în care legătura dintre Mars și Jupiter este în Area 0 are ca efect emiterea repetată a următorului mesaj de syslog:

```
*Mar  1 14:04:46.824: %OSPF-4-ERRRCV: Received invalid packet: mismatch
area ID, from backbone area must be virtual-link but not found from
172.20.0.1, Serial1/0
```

De asemenea, lipsa unei legături virtuale împiedică ruterele din Area 0, definită între Mars și Jupiter, să comunice LSA-uri cu restul sistemului autonom. În consecință, tabela de rutare a lui Jupiter va afișa doar rutele statice, direct conectate, precum și ruta direct conectată a lui Mars:

```
Jupiter(config-if)#do sh ip route
Gateway of last resort is not set
     172.20.0.0/30 is subnetted, 1 subnets
O IA    172.20.0.0 [110/65] via 192.168.0.1, 00:00:07, FastEthernet0/0
C    192.168.0.0/24 is directly connected, FastEthernet0/0
     133.33.0.0/24 is subnetted, 1 subnets
S       133.33.33.0 is directly connected, Null0
```

După configurarea legăturii virtuale, tabela de rutare a lui Jupiter afișează rutele comunicate din toate ariile:

```
Jupiter#sh ip route
Gateway of last resort is 192.168.0.1 to network 0.0.0.0
     99.0.0.0/32 is subnetted, 1 subnets
O E1    99.99.99.99 [110/165] via 192.168.0.1, 00:00:33, FastEthernet0/0
     172.20.0.0/30 is subnetted, 1 subnets
O IA    172.20.0.0 [110/65] via 192.168.0.1, 00:00:33, FastEthernet0/0
     172.30.0.0/30 is subnetted, 1 subnets
O       172.30.0.0 [110/130] via 192.168.0.1, 00:00:33, FastEthernet0/0
     10.0.0.0/24 is subnetted, 1 subnets
O       10.0.0.0 [110/66] via 192.168.0.1, 00:00:33, FastEthernet0/0
C    192.168.0.0/24 is directly connected, FastEthernet0/0
     133.33.0.0/24 is subnetted, 1 subnets
S       133.33.33.0 is directly connected, Null0
O*E2 0.0.0.0/0 [110/1] via 192.168.0.1, 00:00:33, FastEthernet0/0
O IA 202.1.96.0/22 [110/66] via 192.168.0.1, 00:00:33, FastEthernet0/0
O IA 201.1.32.0/20 [110/67] via 192.168.0.1, 00:00:34, FastEthernet0/0
```

O observație importantă în acest caz este faptul că Jupiter vede acum ruta spre rețeaua 10.0.0.0, din Area 0 dintre Mercury, Venussi Earth, ca fiind anunțată prin LSA -uri de tip 2. Datorită legăturii virtuale, Jupiter are acum interfața FastEthernet0/0 conectată direct la Area 0. Segmentele ariei 0 conectate de legături virtuale formează o singură arie continuă de backbonesi nu generează rute inter-area.

```
Jupiter(config-if)#do sh ip ospf data | begin Net
            Net Link States (Area 0)

Link ID         ADV Router      Age         Seq#       Checksum
10.0.0.1        202.1.99.1      51    (DNA) 0x80000018 0x00A44F
192.168.0.2     192.168.0.2     1839        0x80000001 0x009EE
```

6 IS-IS

6.1 Componentele protocolului

6.1.1 ISO vs. IETF

La începutul anilor `80 succesul DARPA a dus la dezvoltarea a numeroase rețele de calculatoare, mai ales în jurul universităților americane. Trecerea de la o infrastructură de comunicație ce trebuia să asigure conectivitate pentru câteva noduri (precum în cazul DARPA), la o rețea cu sute de mii de participanți a adus în discuție problema standardizării.

Soluția propusă de IETF a fost stiva de protocoale TCP/IP. În cadrul acestui model, IETF a propus mai întâi două protocoale de rutare: un protocol bazat pe vectori de distanță, RIP şi un protocol bazat pe starea conexiunilor, OSPF. Extinderea Internetului spre o rețea de sute de milioane de participanți a făcut necesară introducerea unui nou nivel de ierarhizare a informațiilor de rutare, prin gruparea în sisteme autonome, şi prin protocolul BGP.

O soluție istoric anterioară modelului IETF a fost propusă de ISO, prin stiva de protocoale OSI. Protocolul rutat din cadrul acestui model este CLNP, iar protocolul de rutare este ISIS. În SUA, pentru mai bine de 15 ani, toate sistemele din instituțiile guvernamentale trebuiau să fie compatibile ISO, cu alte cuvinte, să ofere suport pentru CLNP şi ISIS. Cu toate acestea, succesul de care s-au bucurat standardele IETF, au limitat numărul implementărilor de ISIS.

ISIS oferă o schemă de adresare ierarhică, cu un spațiu de adrese mult mai mare decât cel oferit de IPv4. Overheadul impus de schimbul informațiilor de rutare este redus (mai scăzut decât în cazul OSPF). Astfel, prin extinderea suportului pentru adresarea IPv4, şi prin standardizarea IETF, ISIS a devenit un concurent puternic pentru OSPF. Varianta standardizată, cu suport pentru IP, poartă denumirea de **Integrated ISIS**, iar pe parcursul acestei cărți, prin ISIS ne vom referi varianta extinsă pentru integrarea cu adresarea IPv4.

Deşi mult mai popular în SUA, există numeroase implementări de ISIS, în nucleu unor furnizori de servicii Internet din România. Una dintre cele mai impresionante soluții ISIS este rețeaua de peste 1000 de rutere din Vodafone România.

6.1.2 Rutarea cu IS-IS

Protocoalele de rutare sunt grupate în două categorii: protocoale de rutare în interiorul sistemului autonom (IGP), şi protocoale inter-AS, denumite generic EGP. ISIS a fost proiectat să ofere reguli pentru publicarea informațiilor de rutare atât între echipamente aflate în cadrului aceluiaşi AS, cât şi pentru sisteme aflate în sisteme autonome diferite. Mai mult chiar, ISIS stabileşte

reguli pentru prin care ruterul de ieşire din rețeaua locală poate fi cunoscut de stațiile din LAN. Cu alte cuvinte, într-o rețea ISO, conectivitatea *end-to-end* se realizează doar printr-un singur protocol, ce îndeplineşte trei obiective distincte: rutarea inter-AS, intra-AS şi în LAN.

Rutarea inter-domenii, sau rutarea inter-AS, este realizată de un proces separat. Acest proces poartă numele de **rutare de nivel 3**.

Pentru asigurarea rutării ierarhice în cadrul aceluiași sistem autonom, ISIS grupează sistemele în arii, folosind două procese diferite de rutare. **Rutarea de nivel 1** oferă conectivitate intra-area. Conectivitatea tuturor ariilor se realizează prin procesul de **rutare de nivel 2**.

În terminologia ISO, stațiile poartă denumirea de *end-system* (ES), iar ruterele sunt numite *intermediate-system* (IS), deci operațiile de rutare sunt realizate exclusiv de către IS-uri. Astfel, conectivitatea la nivelul rețelei locale este realizată de o componentă a protocolului ISIS, denumită ES-IS. ES-IS nu își găseşte corespondentul în Integrated ISIS, în rețelele IPv4 funcția aceasta fiind realizată de protocolul ARP.

În figura 6-1 sunt prezentate cele patru procese de rutare folosite de ISIS. Integrated ISIS a păstrat neschimbate procesele de rutare de nivel 1 şi 2. Rutarea de nivel 3, bazată pe IDRP (*Inter-Domain Routing Protocol*) a fost abandonată în favoarea BGP, iar rutarea de nivel 0 a fost înlocuită de ARP (în rețele Ethernet).

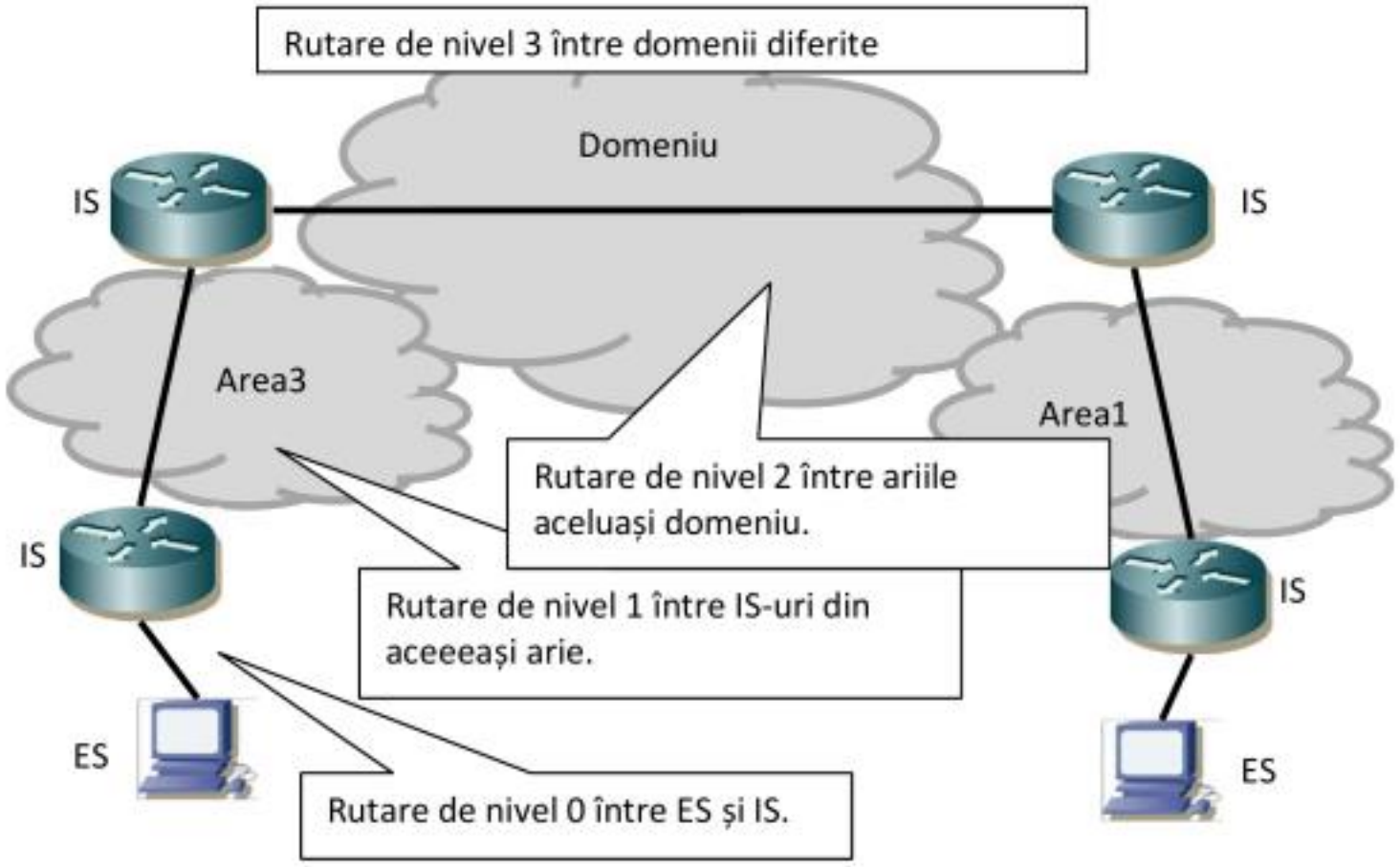

Figura 6-1: Rutarea în ISIS

6.1.3 IS-IS vs. OSPF

Pentru mulți dintre furnizorii de servicii Internet, dorința implementării unei soluții bazate pe un protocol neproprietar se traduce în alegerea dintre ISIS şi OSPF. Din acest motiv acest subcapitol va încerca să pună în balanță avantajele şi limitările celor două protocoale.

Numeroase caracteristici sunt comune celor două protocoale, până la nivelul la care ISIS are mai multe similarități decât diferențe față de OSPF. În terminologia OSPF, ISIS poate fi descris ca fiind *„OSPF ce foloseşte doar arii totally stubby"*. De asemenea, ca şi restul protocoalelor moderne de rutare, ambele suportă VLSM şi oferă o convergență rapidă.

OSPF şi ISIS partajează în primul rând proprietățile protocoalelor *link-state*: de la actualizările incrementale, generate la apariția unor modificări de topologie, până la păstrarea a trei tabele distincte: de adiacențe, de topologie şi de rutare.

Tabela de adiacență va fi creată şi menținută în ambele protocoale pe baza schimbării de pachete de *Hello*. După stabilirea adiacențelor, vor fi trimise pachete de actualizare între vecini, acestea fiind confirmate atât în OSPF, cât şi în ISIS. Pe baza actualizărilor se creează **tabela de topologie**, ce va conține toate informațiile de rutare din cadrul rețelei. Pentru selectarea celor mai bune căi, ambele protocoale vor construi arborele minim de acoperire folosind algoritmul Dijkstra SPF. Cele mai bune rute determinate în urma acestui proces vor fi păstrate în **tabela rutare**.

În cadrul fiecărei rețele multiacces (precum rețelele Ethernet) atât OSPF, cât şi ISIS, vor alege un ruter responsabil cu menținerea consistenței informațiilor de rutare. Acest ruter se numeşte în OSPF *Designated Router* (DR), iar în ISIS *Designated IS* (DIS). OSPF, spre deosebire de ISIS, va mai alege şi un ruter de siguranță ce poate prelua funcțiile DR-ului când acesta nu mai este accesibil: *Backup Designated Router* (BDR), dar în ISIS nu există un echivalent pentru un astfel de ruter.

Cea mai relevantă asemănare între cele două protocoale este **organizarea ierarhică**, în arii, a informațiilor de rutare. Implementarea ariilor este totuşi diferită între OSPF şi ISIS. În OSPF, ruterele sunt situate la granițele dintre arii, astfel că o legătură între două rutere poate aparține doar unei singure arii la un moment dat. În ISIS, un ruter, împreună cu toate interfețele sale, poate face parte doar dintr-o sigură arie, iar capetele unei conexiuni se pot afla în arii ISIS diferite, astfel că granițele dintre arii se află pe legăturile dintre rutere.

Modelul de ierarhizare a ariilor din OSPF, în care aria sa centrală (area 0) interconectează toate celelalte arii, a fost împrumutat şi în ISIS, care introduce conceptul de „*backbone area*", formată din rutere ce pot transporta rute şi pachete între arii diferite ale topologiei ISIS, asemenea ABR-urilor din OSPF.

Sumarizarea adreselor pentru ambele protocoale se va realiza doar între arii, spre deosebire de EIGRP unde sumarizarea se poate realiza, la nivel de interfață, în orice punct al rețelei.

Similar cu OSPF, ISIS suportă *load balancing* între maximum 6 rute spre aceeaşi destinație. Rutele trebuie să aibă aceeaşi distanță administrativă şi acelaşi cost (suma metricilor). ISIS nu suportă *load balancing* pentru rute de cost inegal.

Comparând eficiența celor două soluții, trebuie analizat numărul de pachete de actualizare, precum şi *overhead*-ul implicat de procesul de încapsulare. Principalul avantaj al ISIS îl reprezintă **numărul scăzut de pachete de actualizare** (LSP) trimise între vecini, cel mai adesea, mult mai mic decât în cazul unei implementări OSPF. În termeni de încapsulare, Integrated ISIS rulează direct peste nivelul legătură de date, spre deosebire de OSPF ce rulează peste IP.

Reversul medaliei îl reprezintă puternica integrare a OSPF cu tehnologiile IPv4, larga sa răspândire, în termeni de echipamente, personal calificat şi informații disponibile.

Metrica folosită de OSPF este calculată pe baza lățimii de bandă declarate (*bandwidth*), o interfață serială având un cost mult mai ridicat decât o interfață Ethernet. În ISIS, metrica implicită pentru toate interfețele este 10, iar modificarea manuală a costurilor pentru fiecare interfață este singurul mod de a reflecta (cel puțin teoretic) capacitatea unei legături.

Flexibilitatea şi uşurința de extindere a protocolului sunt avantaje deseori citate în favoarea ISIS. Relevanța acestei diferențe este totuşi limitată, atâta vreme cât OSPF răspunde atât cerințelor implementărilor IPv4 prin OSPFv2, cât şi cerințelor IPv6 prin OSPFv3.

6.1.4 Adresarea ISO

Protocolul de la nivelul rețea propus de ISO este CLNP (*Connectionless Network Protocol*) şi, în genereal, oferă aceleaşi servicii nivelului transport ca şi protocolul IP. Datagramele CLNP sunt procesate în funcţie de antet, suportă fragmentare, un echivalent TTL, opțiuni suplimentare, precum şi alte similitudini cu protocolul IP. Serviciile asigurate prin intermediul protocolului CLNP sunt comasate în CLNS, sau *Connectionless Network Service*.

Inițial, ISIS a fost conceput pentru a funcționa exclusiv în rețele cu trafic CLNS. Apariția Integrated ISIS a extins acoperirea ISIS si asupra protocolului IP. Deşi Integrated ISIS este, în continuare, capabil să ruteze trafic CLNS, el transportă, în prezent, doar trafic IP. Cu toate acestea, ISIS foloseşte în continuare adresele CLNS drept identificatoare pentru rutere, ceea ce duce la crearea tabelelor de topologie interne protocolului bazate pe adrese CLNS. În consecință, o topologie funcțională ISIS presupune configurarea adreselor CLNS pe fiecare ruter (IS), dar acest lucru se face cu scop pur administrativ, fără a influența rutarea pachetelor IP sau propagarea actualizărilor de rutare.

Adresele ISO (CLNS) se prezintă în două forme, în funcție de dispozitivul care este adresat:

- *Network Service Access Point* (NSAP)
- *Network Entity Title* (NET)

O adresă CLNS poate avea de la 8 la 20 de octeți (adresele IP au doar 4) şi cuprinde 3 părți:

- **Area**: echivalentul adresei de rețea a unui subnet IP, descrie un grup sau o zonă;
- **ID**: echivalentul zonei de stație (*host*) dintr-o adresă IP, identifică un anumit dispozitiv din membrii grupului;
- **SEL**: la nivel de stație, câmpul SEL identifică un proces din stația respectivă, similar unui număr de port TCP sau UDP.

IDP		DSP		
AFI (1 octet)	IDI	HODSP	System ID (1-8 octeți)	NSEL (1 octet)
AREA			ID	SEL

Figura 6-2: Câmpurile unei adrese ISO

Din punctul de vedere al tipului de rutare, adresa este împărțită în zona IDP (*Inter Domain Part*), responsabilă cu rutarea externă, între sisteme autonome şi zona DSP (*Domain Specific Part*), ale cărei informații sunt folosite pentru rutarea în interiorul unui sistem autonom. Informațiile pentru cele două modele de rutare prezente în cadrul unei singure adrese ISO arată faptul că schema de adresare în contextul CLNS este una globală.

Primul octet al adresei, câmpul AFI (*Address Family Identifier*) are ca scop specificarea autorității ce stabileşte formatul adresei, iar câmpul IDI (*Initial Domain Identifier*) indică o anumită organizație în interiorul unui AFI, dar nu intervine în procesul de rutare iar prezența sa e opțională. Cele mai folosite valori pentru AFI sunt:

- 39: cod regional
- 47: cod internațional
- 49: adrese private, nerutate în Internet, asemănătoare adreselor private ale protocolului IP, din RFC 1918.

Câmpul HODSP (*High-Order DSP*) identifică o arie din cadrul unui sistem autonom. Toate ruterele din interiorul unei arii trebuie să aibă acelaşi cod în câmpul de arie din adresele ISO configurate.

Câmpul *System ID* indică un ES sau un IS din interiorul unei arii şi poate avea 6 sau 8 octeți, poate fi specificat manual sau poate fi dedus din adresa MAC sau cea IP. Indiferent de metoda folosită, *System ID*-urile din interiorul unei arii, ale ruterelor ce efectuează rutare de nivel 1, trebuie să fie unice. De asemenea, ruterele ce efectează rutare de nivel 2 trebuie să aibă *System ID-uri* unice între toate ruterele de nivel 2.

Ultimul câmp, NSEL, are un octet şi identifică serviciul căruia îi este adresat pachetul. Valoarea câmpului NSEL stabileşte tipul de adresei ISO. Astfel, dacă NSEL are valoarea 0x00, adresa ISO este de tip NET iar acest tip de adresă este folosit pentru a identifica rutere (IS-uri). Orice altă valoare face ca adresa ISO să fie de tip NSAP, utilizabilă de către orice alt dispozitiv sau, în general, de ES-uri.

Din moment ce un ruter, împreună cu toate interfețele sale, poate să facă parte doar dintr-o singură arie, adresele ISO se asignează la nivel de ruter şi nu de interfață. O implementare clasică de ISIS nu necesită mai mult de o adresă ISO de tip NET configurată pe fiecare ruter, dar pot fi configurate până la un maxim de 3 adrese. Adresele multiple sunt folosite în contextul migrării unui ruter dintr-o arie în alta, deci pot diferi prin numărul ariei, dar condiția pentru adrese multiple este ca toate să aibă acelasi *System ID*.

Un exemplu de adresă NET este următorul:

`49.0005.AA00.0301.16CD.00`

Câmpurile adresei pot fi identificate de la dreapta spre stânga astfel:

- Ultimul octet reprezintă câmpul NSEL şi este setat la valoarea 00, ceea ce indică o adresă de ruter (NET).
- Următorii 6 octeți (AA00.0301.16CD) reprezintă *System ID*-ul, ce identifică în mod unic un ruter ISIS.
- Cei 3 octeți rămaşi, de la începutul adresei ISO (49.0005) , reprezintă numărul ariei. Primul octet, cu valoarea 49, indică faptul că adresa este una privată. Datorită adresării private, ce nu impune restricții cu privire la structura adresei, câmpul IDI a putut fi omis, pentru simplificare.

Motivul pentru care o adresă ISO trebuie citită de la dreapta spre stânga se datorează faptului că identificatorul pentru arie poate avea lungimi variabile, în timp ce toate celelalte câmpuri vor fi întotdeauna constante. Spre exemplu, următoarea adresă ISO este una validă:

`49.0015.801f.8ff0.0000.0000.0c00.1234.00`

Pentru acest exemplu, se identifică, de la dreapta spre stânga, primul octet ca fiind câmpul NSEL (00), următorii 6 octeți ca fiind câmpul de *System ID* (`0000.0c00.1234`), iar următorii 9 octeți vor funcționa ca identificator pentru arie (`49.0015.801f.8ff0.0000`).

6.2 Funcționarea IS-IS

6.2.1 Mesaje

Spre deosebire de OSPF, mesajele schimbate între rutere au întotdeauna acelaşi tip, indiferent de conținut. În locul LSA-urilor trimise pentru fiecare informație ce trebuie actualizată, în OSPF, ISIS trimite toate informațiile de rutare într-un singur pachet încapsulat direct în cadrul de la nivelul legătură de date. Pachetele ISIS conțin un antet de 8 octeți urmat de un număr variabil de câmpuri, numite TLV (*Type-Length-Value*) ce descriu informațiile necesare pentru rutare.

Alte două tipuri de *hello*-uri sunt ISH (*Intermediate System Hello*) şi ESH (*End System Hello*). ISH-urile sunt folosite de către IS-uri pentru a se anunța ES-urilor iar ESH-urile sunt folosite de către ES-uri pentru a se anunța ruterelor pe care le vor folosi pe post de *gateway*. Din moment ce stațiile configurate cu protocolul IP nu efectuează rutare de nivel 0, ele nu vor răspunde ISH-urilor, deci

aceste *hello*-uri nu vor fi folosite, iar ruterele vor considera că aceasta la topologie un întreg segment de rețea în locul unor ES-uri particulare.

Informațiile de rutare încapsulate, în ISIS, se numesc LSP-uri (*Link State Packet*) și contin lista adiacențelor și a stărilor vecinilor unui ruter. Deoarece adiacențele se stabilesc separat pentru nivelurile 1și 2, LSP -urile sunt, de asemenea, transmise astfel încât informația de nivel 1 este separată de cea de nivel 2. Astfel, un ruter de nivel 1 va transmite LSP-uri tuturor ruterelor din aria sa, pe când un ruter de nivel 2 va transmite LSP-uri tuturor ruterelor de nivel 2 din domeniul său (și, eventual, de-a lungul mai multor arii).

Pentru sincronizarea bazelor de date între rutere, modificările în topologie sunt anunțate inițial doar ca SNP-uri (*Sequence Number Packet*). Un SNP cuprinde o listă condensată a tuturor LSP-urilor dintr-un ruter și este trimis doar vecinilor direct conectați. La primirea unui SNP, vecinii pot verifica rapid dacă informațiile din SNP sunt mai recente decât cele din propria bază de date și pot cere informații actualizate, ce vor fi primite sub forma unor LSP-uri. Mesajele SNP pot lista întreaga bază de date a stărilor legăturilor unui ruter și se vor numi CSNP (*Complete SNP*) sau pot lista doar un subset al lor, ca în cazul apariției unor modificări izolate în topologie, și se vor numi PSNP (*Partial SNP*).

6.2.2 Adiacențe

Înainte ca doi vecini ISIS să poată schimba rute, ei trebuie să stabilească o adiacență validă. Principalul tip de mesaj ce realizează adiacența între două rutere este mesajul de tip IIH (*Intermediate-to-Intermediate Hello*), ce funcționează asemenea unui protocol de tip " *hello*", întâlnit atât în OSPF cât și în EIGRP. IIH -urile sunt schimbate doar între rutere adiacente și sunt trimise separat pentru nivelurile 1 și 2 de rutare. În consecință, dacă două rutere pot ruta atât intra-area cât și inter-area, ele pot stabili maxim două adiacențe: una la nivelul 1 și alta la nivelul 2.

Deoarece mediile point-to-point și broadcast funcționează diferit, adiacențele se formează diferit. O rețea point-to-point implică doar două noduri ce pot comunica, în timp ce o rețea de tip broadcast este una multiacces, în care pot exista o multitudine de rutere L1 și L2, ceea ce creează necesitatea generării de *hello*-uri pentru fiecare nivel în parte.

Stabilirea adiacențelor pe medii point-to-point se face după ce fiecare parte a primit câte un pachet *hello*. Ruterele schimbă apoi CSNP-uri și își sincronizează bazele de date. Adiacența este menținută apoi un timp nedefinit prin schimb periodic de mesaje *hello*, la fiecare 10 secunde. Dacă de la un vecin nu se primesc mesaje *hello* timp de 30 de secunde, el este considerat deconectat iar din baza de date locală se elimină toate intrările și rutele ce depind de acel vecin.

Mediile broadcast stabilesc un DIS, echivalent al DR-ului din OSPF, responsabil pentru întreaga rețea. Alegerea unui DIS minimizează cantitatea de trafic de rutare ce traversează acel segment și evită situațiile de sincronizare ce nu ar fi necesare între mai rutere multiple responsabile pentru același segment. DIS-ul este singurul ruter responsabil pentru trasnmiterea LSP-urilor ce cuprin informații despre segmentul de rețea respectiv.

DIS-ul creează un pseudonod, asemenea unui ruter virtual care reprezintă întreaga rețea multiacces. *Hello*-urile DIS-ului se trimit de 3 ori mai repede decât cele ale altor rutere, la 3.3 secunde, pentru a detecta rapid erorile. Dacă DIS-ul nu mai poate funcționa, un nou ruter este ales în locul său după criteriul priorității celei mai mari sau, dacă acestea sunt egale, după cel mai mare SNPA (*Subnetwork Point of Attachment*). Pentru segmente Ethernet, SNPA-ul reprezintă valoarea adresei MAC de pe interfață, iar pentru segmente conectate la rețele *Frame Relay*, SNPA-ul este

valoarea identificatorului de circuit de pe interfața respectivă (DLCI). Prioritatea implicită pentru fiecare interfață este 64 şi poate primi valori între 1 şi 127.

ISO descrie cele 4 niveluri de rutare, specificate anterior: nivelul 0, între ES-uri şi IS-uri; nivelul 1, în interiorul unei zone; nivelul 2, între zone diferite; nivelul 3, între sisteme autonome distincte. Integrated IS-IS e responsabil doar pentru nivelurile 1 şi 2. Un ruter ISIS se încadrează într-una dintre categoriile: L1, L2 sau L12, în funcție de tipul sau tipurile de rutare pe care le poate efectua.

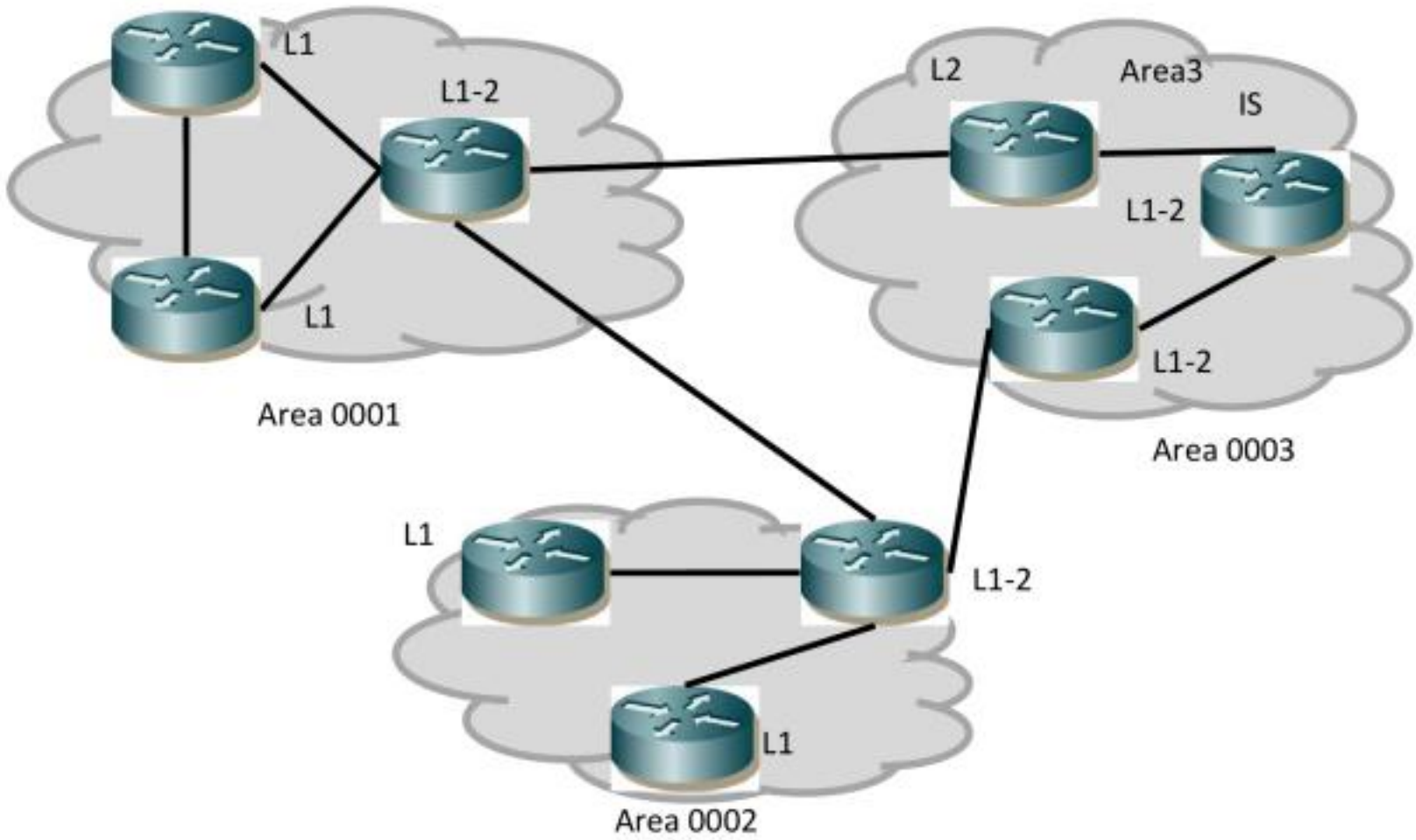

Figura 6-3: Topologie şi zone IS-IS

6.2.3 Backbone L2

Rutarea de nivel 2 se realizează numai între ruterele capabile de rutare de nivel 2 (L2) sau între rutere ce pot ruta atât la nivel 1 cât şi la nivel 2 (L12). Aceste rutere formează zona de backbone într-o topologie ISIS. De regulă, *backbone*-ul se întinde de-a lungul mai multor arii deoarece scopul său este de a realiza rutarea între arii distincte. Orice dispozitiv (IS sau ES) care doreşte să trimită trafic unei destinații dintr-o altă arie decât a sa trebuie să folosească un ruter L2 sau L12 pe post de gateway. Rutarea inter-area, la nivel 2, se face pe baza identificatorilor de arie din adresele NET.

Backbone-ul în sine nu reprezintă o arie separată, ca în cazul OSPF, ci este reprezentat ca un graf de rutere cu cel puțin un nod în fiecare arie din domeniu. În mod automat, fiecare ruter din backbone va face parte dintr-o arie specifică, stabilită de câmpul Area ID din adresa NET.

Deşi chiar şi un ruter L2 va face parte dintr-o anumită arie, el nu va putea trimite pachetele direct spre o destinație L1 din aria sa deoarece el va menține doar adiacențe de nivel2, cu ruterele L2 şi L12 şi îşi va construi corespunzător baza de date din informațiile de nivel 2 ale acestora.

6.2.4 Funcționare L1

Un ruter L1 va cunoaşte în baza sa de date doar destinațiile din aria sa curentă. De asemenea, el va stabili adiacențe şi se va sincroniza doar cu alte rutere L1 sau L12 din aria sa. Traficul trimis de

astfel de rutere, spre o destinație din aceeași arie, nu necesită rutare la nivel 2. Pachetele pot urma o cale formată doar din rutere L1 (eventual L12) până la destinație.

Ruterele L1 vor trebui să folosească un ruter L12 pentru a putea comunica și cu alte destinații, din afara ariei lor. Ele vor primi din partea ruterelor L12 o rută implicită (*default*) și vor trimite celui mai apropiat ruter L12 toate pachetele destinate altor arii.

În interiorul unei arii, ruterele L1 trebuie să aibă același număr de arie și numere distincte de *System ID* pentru a putea identifica dacă destinația unui pachet se află în aceeași arie sau nu, deci dacă trebuie căutată în baza de date cu legături de nivel 1 sau dacă pachetul trebuie trimis la ruterul L12 local.

Protocolul ISIS nu permite ca sumarizarea rutelor să fie făcută în interiorul unei arii, deci ruterele R1 nu sunt capabile de sumarizare.

6.2.5 Funcționarea L12

Un ruter L12 menține două baze de date separate pentru a ține evidența stării legăturilor și topologiilor la ambele niveluri de rutare. El face parte din *backbone*-ul domeniului ISIS și va stabili adiacențe cu toate ruterele L2 și L12 vecine. De asemenea, el va cunoaște atât destinațiile de nivel 2, pentru fiecare arie diferită, cât și destinațiile de nivel 1 ale ariei din care face parte.

Ruterele L12 vor primi pachetele din interiorul ariei lor, pentru a le ruta spre alte arii. Aceste pachete vor fi trimise de către ruterele L12 spre alte rutere din *backbone*, până când acestea ajung la un ruter L12 ce face parte atât din *backbone* cât și din aria destinație.

Sumarizarea ruterelor se face de către ruterele L12 și se aplică rutelor din aria la care este atașat ruterul respectiv. Rutele sumarizate sunt introduse în *backbone* alături de rutele altor rutere L12 și L2. De aceea, dacă există mai mult de un ruter L12 ce oferă acces spre exteriorul unei arii, sumarizarea trebuie configurată pe toate aceste rutere. Altfel, ruterele din backbone vor folosi doar rutele nesumarizate pentru a accesa aria respectivă, deoarece acestea din urma vor fi mai specifice și, deci, preferate în tabelele de rutare.

6.3 Configurarea protocolului

Primul pas în configurarea IS-IS îl reprezintă planificarea schemei de adresare. Spre deosebire de o rețea IP, prin configurarea adreselor ISO va fi stabilit tipul de adiacență ce se poate forma între două rutere. Astfel, pornind de la topologia rețelei vor trebui determinate ariile în care vor fi grupate ruterele, cu alte cuvinte, ruterele ce vor rula doar L1 (cel mai adesea datorită resurselor hardware limitate), L2 (pentru optimizarea funcționării ruterelor de *backbone*) sau L12 (pentru asigurarea conectivității ruterelor L1 la *backbone*).

Ruterele sunt grupate în arii, pentru fiecare arie fiind definit un prefix.

Pentru fiecare ruter, adresa CLNS va cuprinde prefixul (identificatorul ariei), urmat de identificatorul de sistem (unic la nivel de arie) și în final un octet cu valoarea 0. Ultimul câmp din adresa CLNS îl reprezintă NSEL, valoare acestuia fiind 0 pentru adresa unui ruter.

Planificarea adreselor IP nu va avea un impact asupra funcționării IS-IS. Din acest motiv alocarea adreselor IP urmărește în general obținerea unui grad cât mai ridicat de sumarizare.

Setarea adresei va fi realizată în cadrul protocolului de rutare:

```
Cluj(config)#router isis
Cluj(config-router)#net 49.0001.aaaa.aaaa.aaaa.00
```

Pentru a defini interfețele pe care rulează IS-IS va trebui activat protocolul în modul de configurare a interfeței respective:

```
Cluj(config)#interface serial 0/0
Cluj(config-if)#ip router isis
```

Prin activarea protocolului pe o interfață, ruterul va trimite şi va primi actualizări pe respectiva interfață. În plus rețeaua din care face parte interfața va fi inclusă în pachetele de actualizare trimise vecinilor.

Pe ruterele Cisco, modul implicit este L12, deci toate ruterele vor stabili câte două adiacențe între ele şi fiecare va fi capabil de rutare atât la nivel 1 cât şi la nivel 2. În general, topologia poate fi lăsată sub forma inițială, formată din rutere L12, dar, deşi acest lucru simplifică efortul de configurare, rețeaua va fi de două ori mai încărcată din cauza *hello*-urilor şi mesajelor de rutare (LSP-uri şi SNP-uri), iar ruterele vor folosi de două ori mai multă memorie şi putere de procesare, ceea ce nu e întotdeauna o situație acceptabilă într-o rețea activă.

Schimbarea tipului unui ruter se face în modul global de configurare al protocolului de rutare, ca în exemplul următor:

```
Cluj(config)#router isis
Cluj(config-router)#is-type level-1
```

Parametrii acceptați de comanda **`is-type`** sunt: **`level-1`**, **`level-1-2`** şi **`level-2-only`**.

Nivelul de rutare poate fi setat şi în mod particular, pe interfețe, în momentul în care se dorește ca ruterul să stabilească adiacențe diferite în funcție de vecini[1]. În exemplul următor, interfața serială 0/0 va putea stabili adiacențe doar cu o interfață de tip L1 sau L12, iar interfața serială 0/1 va putea stabili adiacențe doar cu o interfață de tip L2 sau L12:

```
Cluj(config)#interface serial 0/0
Cluj(config-if)#isis circuit-type level-1
Cluj(config-if)#interface serial 0/1
Cluj(config-if)#isis circuit-type level-2-only
```

Metricile atribuite de către protocolul de rutare fiecărei interfețe au valoarea implicită de 10 şi se calculează independent pentru nivelul 1 şi 2 de rutare. Pentru modificarea metricilor la nivel de interfață se foloseşte comanda **`isis metric`**, urmată de valoarea noii metrici şi tipul adiacenței pentru care se aplică:

```
R1(config)#interface serial1/0
R1(config-if)#isis metric 25 level-1
R1(config-if)#isis metric 30 level-2
```

Metrica setată pe o interfață este singurul parametru folosit la calcularea rutelor ce vor fi introduse în tabela de rutare. Ruterele calculează costul destinațiilor adunând incremental metricile rutelor primite în LSP-uri cu valorile setate pe interfețele respective.

DIS-ul unui segment multiacces este ales pe baza priorității. Implicit, un ruter în ISIS are priritatea 64 pe toate interfețele. Pentru modificarea priorității la o valoare între 1 şi 127 se foloseşte comanda **`isis priority`**, ca în exemplul următor:

```
R1(config)#interface serial1/0
R1(config-if)#ip route isis
R1(config-if)#isis priority 127
```

---

[1] Setarea unei interfețe pentru un anumit mod de rutare este utilă în special pentru ruterele de tip L12. Acestea vor trimite LSP-uri pe interfețele configurate pentru rutare L1 cu un bit (ATT) setat astfel încât ruterele L1 ce vor primi aceste LSP-uri vor considera automat ruterul L12 drept gateway şi vor instala o rută default către acesta, din moment ce tranzitarea între arii se face prin rutere L2 şi L12.

Pentru a realiza sumarizarea rutelor, pe un ruter de tip L12, se foloseşte comanda **summary-address**, urmată de adresa sumarizată şi masca corespunzătoare. Ruterul L12 va anunța altor arii doar ruta sumarizată.

Spre exemplu, dacă un ruter L12 anunță în ISIS următoarele petru rețele cu masca /24, acestea pot fi sumarizate într-o singură rețea cu masca /22:

- 200.99.104.0 /24
- 200.99.105.0 /24
- 200.99.106.0 /24
- 200.99.107.0 /24

Pentru sumarizarea rețelelor de mai sus, se folosește comanda:

```
Cluj(config)#router isis
Cluj(config-router)#summary-address 200.99.104.0 255.255.255.252
```

Se observă faptul că sumarizarea se execută direct în modul de configurare al protocolului de rutare şi nu pe o interfață specifică.

Ca majoritatea celorlalte protocoale de rutare, ISIS suportă autentificare pentru a nu permite alterarea tabelelor de rutare din partea dispozitivelor ce nu aparțin topologiei ISIS. ISIS permite configurarea de parole pentru fiecare legătura în parte, la nivel de arie şi la nivel de domeniu. Dacă un ruter nu deține parola necesară acesta nu va putea executa operațiile corespunzătoare: nu va putea stabili o adiacență pe un *link*, nu va putea deveni membru al unei arii sau nu va putea deveni membru al unui domeniu de nivel 2.

Autentificarea la nivel de interfață poate fi realizată pentru nivelul 1 sau 2 de rutare, în particular, sau pentru ambele simultan. Dacă nu este specificat nivelul pentru care se face autentificare, se aceasta se aplică pentru ambele niveluri. Parolele sunt transportate în mesajele de tip *hello* ale protocolului, fapt ce separă procesele de autentificare dintre cele două niveluri. Pentru determinarea tipului de adiacență dintre doi vecini se poate folosi comanda **show clns neighbor**. La nivel de arie sau de domeniu autentificarea reprezintă un singur proces comun pentru ambele niveluri.

Exemplu de autentificare la nivel de interfață:

```
R1(config)#interface serial1/0
R1(config-if)#isis password ISISPASSWORD
```

Exemplu de autentificare la nivel de arie:

```
R2(config)#router isis
R2(config-router)#area-password AREAPASS
```

Exemplu de autentificare la nivel de domeniu:

```
R2(config-router)#router isis
R2(config-router)#domain-password DOMAINPASS
```

Pentru funcționarea procesului de autentificare şi, implicit, a protocolului de rutare în contextul autentificat, toate ruterele trebuie să posede informații consistente de autentificare la nivel de legătură (*link*), arie sau domeniu, după caz.

6.3.1 Comenzi de verificare

Trei dintre cele mai utile comenzi pentru verificarea funcționării protocolului ISIS sunt următoarele:

- show clns neighbors

- show clns interface
- show isis database

Comanda show clns neighbors include informații din tabela de vecini a protocolului împreună cu starea legăturior cu aceştia:

```
R1#show clns neighbors

System Id      Interface SNPA            State Holdtime Type Protocol
0000.0000.000D Et0       00e0.1e3d.d56f Up    8        L1   IS-IS
1111.2222.3333 Se1/0     *HDLC*         Up    23       L1L2 IS-IS
```

În exemplul de mai sus, RouterA are doi vecini, identificați prin valoarea *System ID* configurată prin adresele NET de pe fiecare ruter. Câmpul *„interface"* specifică interfața prin care ruterul este conectat la fiecare vecin. Câmpul SNPA (*Subnetwork Point of Attachment*) indică tipul legăturii de nivel 2 pe care ruterul o are cu vecinul respectiv. Pentru rețele Ethernet, SNPA-ul indică adresa MAC a interfeței. Pentru interfața Serial1/0, SNPA indică protocolul point-to-point de pe interfață, HDLC.

Pentru mai multe detalii, ce includ durata pentru care interfața a fost activă, adresa zonei şi adresa IP a vecinului, se poate adăuga parametrul `detail` comenzii:

```
R1#show clns neighbors detail

System Id      Interface   SNPA        State  Holdtime  Type Protocol
R2             Se1/0       *HDLC*      Up     21        L1L2 IS-IS
  Area Address(es): 49.0000
  IP Address(es):  10.1.1.2*
  Uptime: 01:30:00
  NSF capable
```

Comanda **`show clns interface`** afiseaza informatiile specifice fiecărei interfețe ce rulează ISIS. Comanda acceptă ca parametru şi numele unei interfețe pentru a filtrarea rezultatului:

```
R1#show clns interface serial 1/0
Serial1/0 is up, line protocol is up
  Checksums enabled, MTU 1500, Encapsulation HDLC
  ERPDUs enabled, min. interval 10 msec.
  CLNS fast switching enabled
  CLNS SSE switching disabled
  DEC compatibility mode OFF for this interface
  Next ESH/ISH in 23 seconds
  Routing Protocol: IS-IS
    Circuit Type: level-1-2
    Interface number 0x0, local circuit ID 0x100
    Neighbor System-ID: R2
    Level-1 Metric: 25, Priority: 64, Circuit ID: R2.00
    Level-1 IPv6 Metric: 10
    Number of active level-1 adjacencies: 1
    Level-2 Metric: 30, Priority: 64, Circuit ID: R1.00
    Level-2 IPv6 Metric: 10
    Number of active level-2 adjacencies: 1
    Next IS-IS Hello in 8 seconds
    if state UP
```

Protocolul de rutare indicat (**`IS-IS`**) arată faptul că ruterul a realizat doar adiacențe cu alte rutere. Lipsa rutării de tip CLNS are drept efect şi absența protocolului de rutare ES-IS. **`circuit type`** indică faptul că pe interfața Serial1/0 se încearcă realizarea de adiacențe L1 şi L2, iar realizarea lor este confirmată de numărul 1 din dreptul ambelor tipuri de adiacențe. De asemenea, se poate observa şi faptul că metricile sunt calculate separat pentru fiecare tip de adiacență, ceea ce duce la calcularea unor tabele de topologie diferite pentru rutarea de tip L1 şi cea de tip L2.

Comanda **`show isis database`** afişează informațiile obținute prin LSP-uri din baza de date locală:

```
R1#show isis database

IS-IS Level-1 Link State Database:
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime      ATT/P/OL
R1.00-00            * 0x00000013   0x723A        701               0/0/0
R2.00-00              0x00000011   0x3497        596               0/0/0
IS-IS Level-2 Link State Database:
LSPID                 LSP Seq Num  LSP Checksum  LSP Holdtime      ATT/P/OL
R1.00-00            * 0x00000015   0x8006        742               0/0/0
R2.00-00              0x00000012   0x5866        677               0/0/0
```

Indentificatorul de sub câmpul LSPID afişează identificatorul ruterului care a originat LSP-ul, urmat de o valoare care este diferită de zero dacă LSP-ul a originat de la un pseudonod şi de o altă valoare care indică dacă LSP-ul a fost fragmentat, dacă este diferită de zero. *Holdtime*-ul indică timpul, în secunde, pentru care LSP-ul respectiv va mai fi valid în memoria ruterului. Bitul ATT este setat de către un ruter L12 atunci când se anunță drept default gateway pentru o zonă L1. Pentru a afişa şi conținutul LSP-urilor (rutele anunțate, împreună cu metricile), se poate adăuga parametrul **`detail`** comenzii.

Câteva dintre comenzile de *debug* ce pot fi folosite pentru urmărirea în timp real a evenimentelor ISIS dintr-un ruter sunt următoarele:

- **`debug isis adj-packets`** – afişează informații legate de evoluția relațiilor de adiacență cu toți vecinii unui ruter, inclusiv pachetele *hello* trimise şi primite, precum şi eventualele schimbări în starea legăturilor.
- **`debug isis spf-events`** – afişează calculele efectuate de algoritmul SPF (*Shortest Path First*) pentru calcularea rutelor spre destinații, împreună cu toate informațiile ce participă la calcularea topologiei, provenite din LSP-uri.
- **`debug isis update-packets`** – afişează CSNP-urile, PSNP-urile şi LSP-urile primite şi emise de ruter.

6.4 Studiu de caz

În următorul scenariu, furnizorul de servicii ScaleMore, observând consumul de memorie de la nivelul ruterelor din rețea, decide migrarea protocolului de rutare de la EIGRP la IS-IS. Datorită metricii inferioare a IS-IS configurarea noului protocol poate fi făcută fără a afecta informațiile de rutare ce ajung în tabela de rutare.

Pentru evaluarea paşilor de configurare a IS-IS în cadrul rețelei ScaleMore departamentul de proiectare decide testarea mecanismelor de propagare a informațiilor de rutare în cadrul unei topologii cu trei rutere, prezentată în Figura 6-4. Testele vor urmării modalitățile de alterare a informațiilor de rutare, precum şi mecanismele de asigurare a autenticității vecinilor IS-IS.

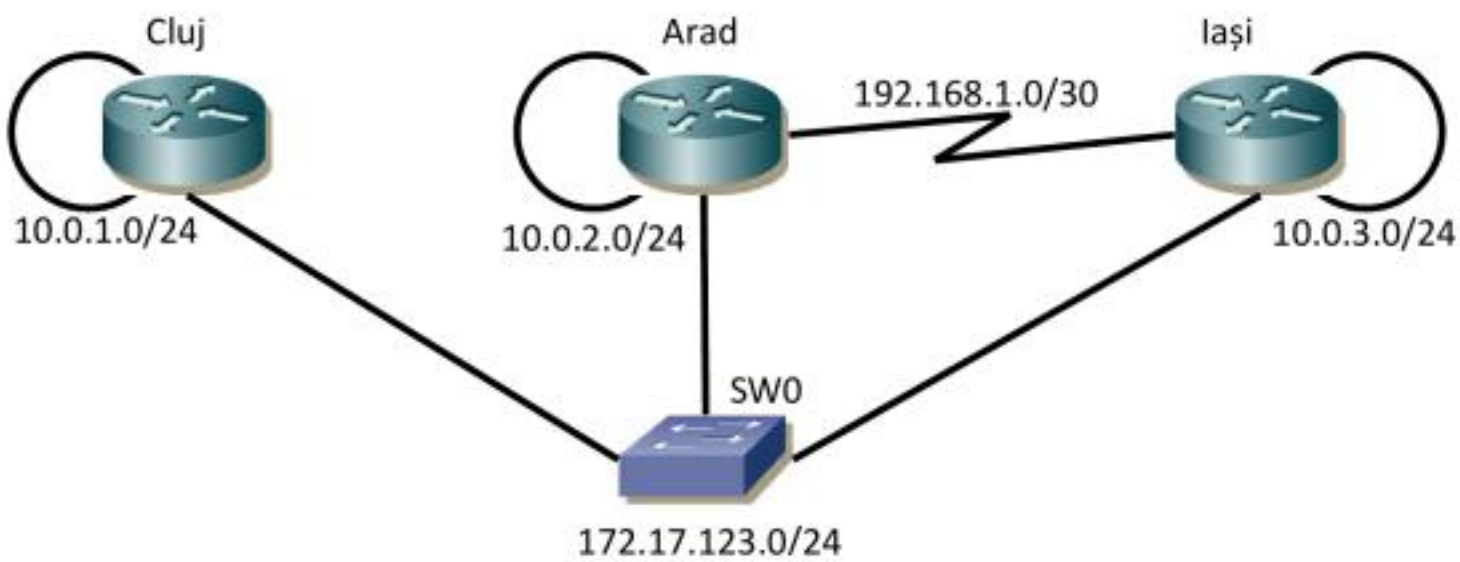

Figura 6-4 Topologia rețelei de test

6.4.1 Configurări de bază

Ruterele fac parte din același domeniu: 49.0001. Cluj va avea *SystemID* aaaa.aaaa.aaaa, Arad bbbb.bbbb.bbbb, iar Iași cccc.cccc.cccc.

Cele trei rutere fac parte din rețeaua FastEthernet 172.17.123.0/24. Între Arad și Iași este configurată o legătură serială ce va primi adrese din spațiul: 192.168.1.0/30. Fiecare rutere va avea configurată o interfață de loopback conform topologiei. Configurația IP a interfețelor este rezumată mai jos:

```
Cluj#show ip interface brief | include up
FastEthernet0/0            172.17.123.1    YES manual up                    up
Loopback0                  10.0.1.1        YES manual up                    up

Arad#show ip interface brief | include up
FastEthernet0/0            172.17.123.2    YES manual up                    up
Serial1/0                  192.168.1.1     YES manual up                    up
Loopback0                  10.0.2.1        YES manual up                    up

Iasi#show ip interface brief | include up
FastEthernet0/0            172.17.123.3    YES manual up                    up
Serial1/0                  192.168.1.2     YES manual up                    up
Loopback0                  10.0.3.1        YES manual up                    up
```

Conform informațiilor de domeniu și *System ID*, pe rutere se va activa protocolul de rutare ISIS, se vor include interfețele active în procesul ISIS și vor fi configurate adresele NET ca în exemplul următor:

```
Cluj(config)#router isis
Cluj(config-router)#net 49.0001.aaaa.aaaa.aaaa.00
Cluj(config-router)#interface fastethernet 0/0
Cluj(config-if)#ip router isis
Cluj(config-if)#interface loopback0
Cluj(config-if)#ip router isis

Arad(config)#router isis
Arad(config-router)#net 49.0001.bbbb.bbbb.bbbb.00
Arad(config-router)#interface serial1/0
Arad(config-if)#ip router isis
Arad(config-if)#interface loopback0
Arad(config-if)#ip router isis
Arad(config-if)#interface fastethernet0/0
Arad(config-if)#ip router isis

Iasi(config)#router isis
Iasi(config-router)#net 49.0001.cccc.cccc.cccc.00
Iasi(config-router)#interface serial1/0
Iasi(config-if)#ip router isis
Iasi(config-if)#interface fastethernet0/0
Iasi(config-if)#ip router isis
Iasi(config-if)#interface loopback0
Iasi(config-if)#ip router isis
```

Primul pas pentru verificarea funcționării protocolului de rutare este verificarea stabilirii adiacențelor. Pentru exemplificare, se vor analiza informațiile de pe ruterul Iași:

```
Iasi#show clns neighbors

System Id      Interface   SNPA            State  Holdtime  Type Protocol
Cluj           Fa0/0       cc00.300c.0000  Up     24        L1L2 IS-IS
Arad           Fa0/0       cc01.300c.0000  Up     28        L1L2 IS-IS
Arad           Se1/0       *HDLC*          Up     20        L1L2 IS-IS
```

Conform rezultatului de mai sus, se observă faptul că ruterul Iași a stabilit adiacențe de tip L1L2 cu vecinii săi deoarece toate ruterele, în mod implicit, rutează atât la nivel 1 cât și la nivel 2. Pentru

ruterul Iaşi, ruterul Cluj este accesibil prin intermediul legăturii multiacces FastEthernet, la care este conectat cu interfața FastEthernet0/0. Ruterul Arad este accesibil atât prin rețeaua FastEthernet (interfața FastEthernet0/0) cât şi prin legătura serială directă (interfața Serial1/0) dintre ruterele Arad şi Iaşi.

Adăugând parametrul detail comenzii anterioare, rezultatul informează asupra numărului ariei fiecărui vecin, adresei IP a interfeței ce a stabilit adiacența şi durata de viață a adiacenței:

```
Iasi#show clns neighbors detail

System Id      Interface   SNPA            State  Holdtime  Type Protocol
Cluj           Fa0/0       cc00.300c.0000  Up     28        L1L2 IS-IS
  Area Address(es): 49.0001
  IP Address(es):  172.17.123.1*
  Uptime: 00:01:08
  NSF capable
Arad           Fa0/0       cc01.300c.0000  Up     24        L1L2 IS-IS
  Area Address(es): 49.0001
  IP Address(es):  172.17.123.2*
  Uptime: 00:01:08
  NSF capable
Arad           Se1/0       *HDLC*          Up     25        L1L2 IS-IS
  Area Address(es): 49.0001
  IP Address(es):  192.168.1.1*
  Uptime: 00:01:15
  NSF capable
```

Se recomadă consultarea relațiilor de adiacență şi pe celelalte rutere:

```
luj#show isis neighbors

System Id      Type Interface IP Address      State Holdtime Circuit Id
Arad           L1   Fa0/0     172.17.123.2    UP    24       Iasi.01
Arad           L2   Fa0/0     172.17.123.2    UP    23       Iasi.01
Iasi           L1   Fa0/0     172.17.123.3    UP    9        Iasi.01
Iasi           L2   Fa0/0     172.17.123.3    UP    8        Iasi.01

Arad#show isis neighbors

System Id      Type Interface IP Address      State Holdtime Circuit Id
Cluj           L1   Fa0/0     172.17.123.1    UP    26       Iasi.01
Cluj           L2   Fa0/0     172.17.123.1    UP    29       Iasi.01
Iasi           L1   Fa0/0     172.17.123.3    UP    7        Iasi.01
Iasi           L2   Fa0/0     172.17.123.3    UP    8        Iasi.01
Iasi           L1L2 Se1/0     192.168.1.2     UP    23       00
```

Comanda **`show isis neighbors`** listează separat fiecare tip de adiacență realiată cu fiecare vecin. Se observă că, pentru segmentul FastEthernet, între fiecare două rutere s-a stabilit câte o pereche de adiacențe deoarece mesajele Hello se trimit independent pentru cele două niveluri de rutare. Pentru legătura serială se trimite un singur tip de mesaj Hello ce stabileşte adiacența necesară.

De asemenea, din rezultatul comenzilor show clns neighbors şi show isis neighbors se deduce faptul că ruterul Iaşi este DIS pentru segmentul FastEthernet. Iaşi are cea mai mare adresă MAC (valoarea SNPA) pe segmentul respectiv iar valoarea **`Iasi.01`** a identificatorului de circuit arată că acesta este pseudonodul segmentului multiacces.

În acest moment, tabelele de rutare trebuie să conțină rute spre toate rețelele cu interfețe active din topologie:

```
Cluj#show ip route isis
     10.0.0.0/24 is subnetted, 3 subnets
```

```
i L1     10.0.2.0 [115/20] via 172.17.123.2, FastEthernet0/0
i L1     10.0.3.0 [115/20] via 172.17.123.3, FastEthernet0/0
     192.168.1.0/30 is subnetted, 1 subnets
i L1     192.168.1.0 [115/20] via 172.17.123.2, FastEthernet0/0
                     [115/20] via 172.17.123.3, FastEthernet0/0

Arad#show ip route isis
     10.0.0.0/24 is subnetted, 3 subnets
i L1     10.0.3.0 [115/20] via 172.17.123.3, FastEthernet0/0
                  [115/20] via 192.168.1.2, Serial1/0
i L1     10.0.1.0 [115/20] via 172.17.123.1, FastEthernet0/0

Iasi#show ip route isis
     10.0.0.0/24 is subnetted, 3 subnets
i L1     10.0.2.0 [115/20] via 172.17.123.2, FastEthernet0/0
                  [115/20] via 192.168.1.1, Serial1/0
i L1     10.0.1.0 [115/20] via 172.17.123.1, FastEthernet0/0
```

Din moment ce metricile nu au fost încă alterate şi sunt constante pentru toate tipurile de interfețe, calcularea rutelor în ISIS este echivalentă cu determinarea numărului de hop-uri până la destinație realizată de protocolul RIP, ceea ce determină introducerea de rute alternative, cu distanțe administrative şi metrici egale în tabela de rutare.

6.4.2 Controlul procesului de alegere DIS

Rezultatul comenzii show clns interface executată pe ruterul Iaşi arată că acesta a fost ales DIS pentru segmentul de rețea FastEthernet (Circuit ID are valoarea **`Iasi.01`**):

```
Iasi#show clns interface fast0/0
FastEthernet0/0 is up, line protocol is up
  Checksums enabled, MTU 1497, Encapsulation SAP
  ERPDUs enabled, min. interval 10 msec.
  CLNS fast switching enabled
  CLNS SSE switching disabled
  DEC compatibility mode OFF for this interface
  Next ESH/ISH in 35 seconds
  Routing Protocol: IS-IS
    Circuit Type: level-1-2
    Interface number 0x1, local circuit ID 0x1
    Level-1 Metric: 10, Priority: 64, Circuit ID: Iasi.01
    Level-1 IPv6 Metric: 10
    Number of active level-1 adjacencies: 2
    Level-2 Metric: 10, Priority: 64, Circuit ID: Iasi.01
    Level-2 IPv6 Metric: 10
    Number of active level-2 adjacencies: 2
    Next IS-IS LAN Level-1 Hello in 1 seconds
    Next IS-IS LAN Level-2 Hello in 715 milliseconds
```

Procedeul de alegere al unui DIS pentru un segment multiacces compară întâi valorile priorităților configurate pe fiecare interfață a unui ruter conectată la segmentul respectiv. În cazul de față, prioritățile tuturor interfețelor sunt setate la valorile implicite, de 64, deci alegerea s-a făcut după valoarea SNPA (adresa MAC) cea mai mare de pe segment, ruterul Iasi câştigând rolul de DIS.

Prioritatea unei intefețe pentru un segment multiacces poat fi modificată direct din modul de configurare al interfeței, ca în exemplul următor:

```
Arad(config)#int fast 0/0
Arad(config-if)#isis priority ?
  <0-127>  Priority value

Arad(config-if)#isis priority 100
```

Comanda anterioară schimbă prioritatea interfeței FastEthernet a ruterului Arad la o valoare mai mare de 64. În consecință, comanda **show clns interface** va indica pe fiecare ruter noul DIS:

```
Iasi#show clns interface fast0/0
FastEthernet0/0 is up, line protocol is up
  Checksums enabled, MTU 1497, Encapsulation SAP
  ERPDUs enabled, min. interval 10 msec.
  CLNS fast switching enabled
  CLNS SSE switching disabled
  DEC compatibility mode OFF for this interface
  Next ESH/ISH in 2 seconds
  Routing Protocol: IS-IS
    Circuit Type: level-1-2
    Interface number 0x1, local circuit ID 0x1
    Level-1 Metric: 10, Priority: 64, Circuit ID: Arad.02
    Level-1 IPv6 Metric: 10
    Number of active level-1 adjacencies: 2
    Level-2 Metric: 10, Priority: 64, Circuit ID: Arad.02
    Level-2 IPv6 Metric: 10
    Number of active level-2 adjacencies: 2
    Next IS-IS LAN Level-1 Hello in 7 seconds
    Next IS-IS LAN Level-2 Hello in 8 seconds
```

Comanda **isis priority** urmată de un parametru numeric schimbă prioritatea unei interfețe pentru ambele niveluri de rutare. Pentru o configurare mai granulară, prioritatea poate fi schimbată doar pentru un anumit nivel de rutare, ca în exemplul următor:

```
Cluj(config)#interface fastethernet0/0
Cluj(config-if)#isis priority 125 ?
  level-1  Specify priority for level-1 routing
  level-2  Specify priority for level-2 routing
  <cr>

Cluj(config-if)#isis priority 125 level-1
```

Ruterului Cluj i-a fost modificată prioritatea pentru nivelul 1 de rutare de la valoarea implicită, de 64, la cea de 125. Drept urmare, noua valoare pentru nivelul 1 este mai mare decât cea de 100 configurată pe ruterul Arad iar ruterul Cluj devine DIS pentru nivelul 1 de rutare pentru segmentul FastEthernet:

```
FastEthernet0/0 is up, line protocol is up
  Checksums enabled, MTU 1497, Encapsulation SAP
  ERPDUs enabled, min. interval 10 msec.
  CLNS fast switching enabled
  CLNS SSE switching disabled
  DEC compatibility mode OFF for this interface
  Next ESH/ISH in 4 seconds
  Routing Protocol: IS-IS
    Circuit Type: level-1-2
    Interface number 0x1, local circuit ID 0x1
    Level-1 Metric: 10, Priority: 64, Circuit ID: Cluj.01
    Level-1 IPv6 Metric: 10
    Number of active level-1 adjacencies: 2
    Level-2 Metric: 10, Priority: 64, Circuit ID: Arad.02
    Level-2 IPv6 Metric: 10
    Number of active level-2 adjacencies: 2
    Next IS-IS LAN Level-1 Hello in 850 milliseconds
    Next IS-IS LAN Level-2 Hello in 435 milliseconds
```

Informațiile cu privire la DIS-ul şi pseudonodul corespunzător unui segment sunt vizibile şi consistente pentru orice interfață conectată la segmentul multiacces respectiv.

6.4.3 Alterarea metricilor

ISIS foloseşte o metrică implicită şi constantă la valoarea 10 pentru toate interfețele. În momentul în care un ruter primeşte o rută pe o anumită interfață, valoarea metricii de pe interfața respectivă este adunată la metrica deja primită, iar noua valoare intră în calculele pentru tabela de rutare.

Analizând tabela de rutare a ruterului Cluj se observă că acesta primeşte două rute cu aceeaşi distanță administrativă şi aceeaşi metrică spre destinația 192.168.1.0/30 (legătura punct la punct dintre Arad şi Iaşi) şi le introduce pe ambele în tabela de rutare. Cele două rute provin de la ruterele Arad şi Iaşi, capetele segmentului.

```
Cluj#show ip route isis | begin 192.168.1.0
     192.168.1.0/30 is subnetted, 1 subnets
i L1    192.168.1.0 [115/20] via 172.17.123.2, FastEthernet0/0
                    [115/20] via 172.17.123.3, FastEthernet0/0
```

Dacă, spre exemplu, se doreşte doar utilizarea rutei prin Iasi se poate altera metrica de pe interfața serială a ruterului Iasi la o valoare mai mică decât cea implicită, pentru ca ruterul Cluj să primească ruta de la Iasi cu un cost mai mic decât cea de la Arad şi să o introducă doar pe aceasta în tabela de rutare:

```
Iasi(config)#interface serial1/0
Iasi(config-if)#isis metric 5 level-1
```

După câteva secunde, tabela de rutare de pe ruterul Cluj pentru destinația 192.168.1.0/30 va arăta o singură destinație:

```
Cluj#show ip route isis | begin 192.168.1.0
     192.168.1.0/30 is subnetted, 1 subnets
i L1    192.168.1.0 [115/15] via 172.17.123.3, FastEthernet0/0
```

Costul inițial de 20 era dat de suma metricilor de pe interfața Serial1/0 a ruterului Iasi şi interfața FastEthernet0/0 a ruterului Cluj. Modificarea costului de pe interfața serială a ruterului Iasi la valoarea 5 are ca efect introducerea în tabela de rutare a ruterului Cluj a unei rute de cost 15.

6.4.4 Autentificarea în IS-IS

Autentificarea mesajelor schimbate între rutere are drept scop protejarea informațiilor de rutare împotriva eventualilor atacatori ce pot introduce rute nevalide în topologie.

La nivelul unei segment de rețea, autentificarea se face direct pe interfețele conectate la segmntul respectiv:

```
Cluj(config)#interface fastethernet0/0
Cluj(config-if)#isis password CISCO
*Mar  1 02:40:29.707: %CLNS-4-AUTH_FAIL: ISIS: LAN IIH authentication
failed
```

Configurarea autentificării pe interfața FastEthernet0/0 a ruterului Cluj rezultă într-un mesaj syslog ce anunță că autentificare eşuează. Eroarea se datorează faptului că niciun alt membru al segmentului nu a fost configurat pentru autentificare. Eşuarea autentificării la nivelul unui segment duce la imposibilitatea stabilirii complete a adiacențelor. Starea ruterului Cluj este văzută ca fiind INIT, şi nu UP, pentru ceilalți doi vecini ai săi:

```
Arad#show clns neighbors

System Id      Interface   SNPA            State  Holdtime  Type
Protocol
Cluj           Fa0/0       cc00.300c.0000  Init   29        L1L2 IS-IS
```

```
Iasi            Fa0/0       cc02.300c.0000      Up      28         L1L2 IS-IS
Iasi            Se1/0       *HDLC*              Up      22         L1L2 IS-IS
```

În consecință, rutele învățate de la ruterul Cluj sunt eliminate din tabelele de rutare ale vecinilor. Se observă că rețeaua interfeței de loopback a ruterului Cluj lipseşte din tabela celorlaltor rutere:

```
Iasi#show ip route isis
     10.0.0.0/24 is subnetted, 2 subnets
i L1    10.0.2.0 [115/15] via 192.168.1.1, Serial1/0
```

Autentificarea la nivel de interfață poate fi particularizată pentru un anumit nivel de rutare prin comanda **isis password CISCO level-1** sau **isis password CISCO level-2**. Autentificarea incorectă pentru un anumit nivel de rutarea poate fi verificată prin comanda **show isis neighbors**, care arată starea adiacențelor de nivel 1 şi 2 pentru fiecare vecin.

În plus, autentificarea poate fi realizată la nivel de arie şi la nivel de domeniu. Configurarea completă a tuturor celor trei tipuri de autentificare este acoperită în următoarea secvență:

```
Arad(config)#interface fastethernet0/0
Arad(config-if)#isis password CISCO

Iasi(config)#interface fastethernet0/0
Iasi(config-if)#isis password CISCO

Cluj(config)#router isis
Cluj(config-router)#area-password AREAPASS
Cluj(config-router)#domain-password DOMPASS

Arad(config)#router isis
Arad(config-router)#area-password AREAPASS
Arad(config-router)#domain-password DOMPASS

Iasi(config)#router isis
Iasi(config-router)#area-password AREAPASS
Iasi(config-router)#domain-password DOMPASS
```

7 Optimizarea rutării

Până în anul 2000 majoritatea rutării din Internet era făcută prin intermediul rutelor statice. Față de protocoalele de rutare dinamice, acestea nu aveau probleme de convergență sau de securitate, singurul lor dezavantaj fiind efortul administrativ mare de a le întreține. Odată cu anul 2000 Internetul a cunoscut o creştere remarcabilă iar nevoia pentru un mecanism dinamic pentru a propaga rute a devenit evidentă. Protocoalele de rutare încorporau deja posibilitatea de autentificare, iar timpul de convergență foarte bun în cazul protocoalelor link-state le-au promovat în rândul rețelelor Enterprise şi ISP. Singurele cazuri din prezent în care rutarea statică este încă preferată sunt în rețelele foarte mici sau rețelele cu cerințe mari de securitate.

Un protocol de rutare trebuie să fie cât mai eficient în deciziile de rutare şi determinare a căilor. Tehnicile de optimizare a rutării sunt mecanisme ce funcționează „peste" protocolul de rutare şi care modifică comportamentul implicit al direcționării pachetelor în funcție de politicile administratorului.

7.1 Redistribuția între multiple protocoale de rutare

Comunicarea rutelor între diferite protocoale de rutare este unul din cele mai complexe subiecte din rutarea IP. Cu toate că folosirea unui singur protocol de rutare în întreaga rețea a unei companii este întotdeauna soluția preferată şi de asemenea şi cea mai întâlnită, apar câteva cazuri în care trebuie realizată comunicarea rutelor între protocoale de rutare diferite. Această comunicare de rute poartă numele de **redistribuție**. Deşi la nivel de comenzi de configurare redistribuția se face foarte uşor, un administrator de rețea trebuie să cunoască foarte bine teoria - deloc trivială - din spatele procesului. Greşelile ce pot fi făcute în redistribuție dețin recordul (alături de protocolul BGP) la raportul dintre dimensiunea extrem de mică a greşelii şi impactul devastator pe care îl poate avea asupra rețelei.

Nevoia de redistribuție între două protocoale de rutare poate apărea din multiple motive:

- Fuziunea a două sau mai multe companii ce folosesc protocoale diferite;
- Utilizarea unei rețele în care există rutere de la producători diferiți ce rulează protocoale proprietare (ex. rutere Cisco ce rulează EIGRP);
- Utilizarea unei rețele în care unele rutere nu sunt îndeajuns de puternice încât să suporte rularea protocolului dominant (OSPF sau EIGRP) şi sunt forțate către un protocol de rutare mai simplu (RIPv2);

7.1.1 Conceptul de redistribuție

Din punctul de vedere al sistemului de operare IOS, un protocol de rutare rulează ca şi orice alt proces de pe ruter. Pentru că majoritatea protocoalelor de rutare funcționează creând adiacențe între dispozitive diferite de rețea, singura metodă prin care două protocoale de rutare pot schimba date între ele este prin comunicație între procese pe acelaşi sistem de operare/ruter (IPC[1]).

Astfel, la redistribuția între două protocoale de rutare, ambele trebuie să ruleze pe acelaşi ruter care va juca rolul unui punct de redistribuție în rețea.

Redistribuția poate fi definită prin cele două procese de mai jos:

1. Toate rutele din tabela de rutare învățate prin protocolul de rutare **din** care se face redistribuția vor fi anunțate în protocolul **în** care se face redistribuția.
2. Toate rutele rețelelor direct conectate care fac parte **din** protocolul de rutare din care se face redistribuția vor fi anunțate **în** protocolul de rutare **în** care se face redistribuția.

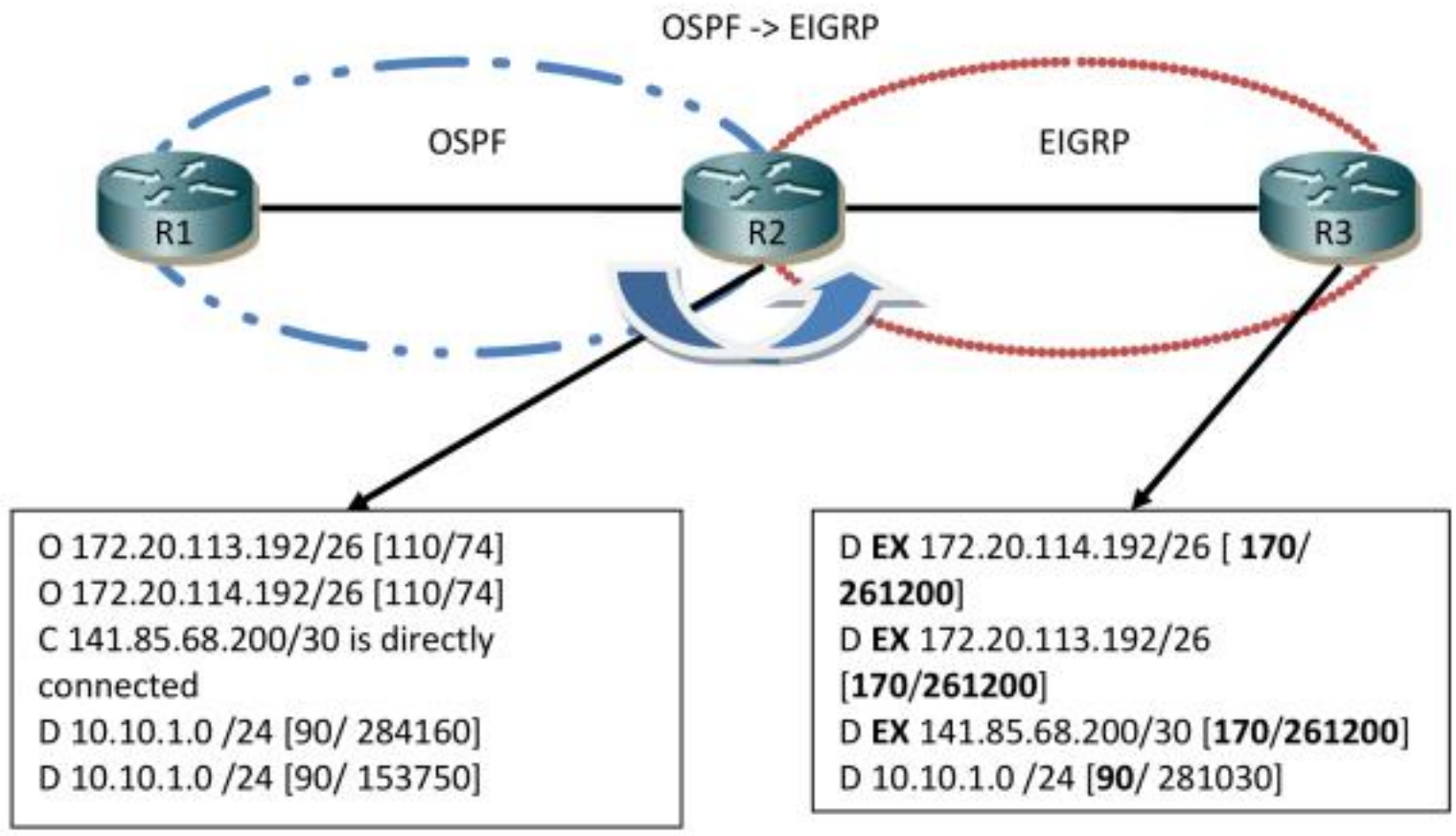

Figura 7-1: Tabelele de rutare după redistribuție

Figura de mai sus prezintă tabelele de rutare ale lui R2 şi R3 în urma redistribuirii unidirecționale OSPF -> EIGRP folosind ruterul R2 ca punct de redistribuție. Există mai multe observații importante ce trebuie realizate pe baza acestui exemplu:

- Tabela de rutare a lui R2 a rămas neschimbată în urma redistribuirii. Acest comportament este normal deoarece R2 avea toate rutele, atât OSPF cât şi EIGRP, în RAM, înaintea schimbului de rute.
- Pentru că ruterul R3 are o adiacență validă cu R2, acesta din urmă i-a comunicat noile rute primite din domeniul OSPF
- Există protocoale de rutare care diferențiază prin distanță administrativă rutele interne de cele externe. Pe R3, protocolul EIGRP setează în acest caz o distanță administrativă de 170. Acest comportament ajută în evitarea rutării suboptimale şi în alegerea căii cele mai bune către aceeaşi destinație cunoscută . Aceste mecanisme vor fi prezentate în următorul subcapitol.
- Toate rețelele externe au intrat în protocolul EIGRP cu aceeaşi metrică. Acest comportament este rezultatul unei pierderi de informații de metrică, inevitabilă la trecerea

[1] http://en.wikipedia.org/wiki/Inter-process_communication

dintr-un protocol în altul. Metricile EIGRP şi OSPF nu sun compatibileşi de aceea EIGRP utilizează o **metrică de intrare** (*seed metric*) pe care o setează oricărei rute externe. Aceasta metrică de intrare este apoi incrementată în interiorul domeniului EIGRP funcţie de parametrii *bandwidth* şi *delay*.

Atenţie! Metrica de intrare nu este setată în mod implicit de către niciun protocol de rutare în afară de OSPF şi trebuie setată manual folosind comanda *default-metric* **sau la redistribuţie. Dacă acest lucru nu este făcut redistribuţia nu va funcţiona.**
Metrica implicită a protocolului OSPF este 20.

În exemplul anterior se realizează o redistribuţie unidirecţională. În funcţie de politica reţelei, acest proces poate fi bidirecţional prin transmiterea rutelor din OSPF în EIGRP şi invers.

Pe lângă redistribuţia între două protocoale de rutare diferite se mai poate realiza şi redistribuirea rutelor statice sau direct conectate într-un protocol de rutare dinamic. În acest caz toate protocoalele folosesc o metrică de intrare implicită, nefiind nevoie ca aceasta să fie specificată manual.

Metricile implicite pentru fiecare protocol de rutare pot fi consultate în tabelul de mai jos:

Protocol sursă	Protocol destinaţie				
	RIP	EIGRP	OSPF	ISIS	BGP (MED)
Connected	1	Metrica int	20 (E2)	0	0
Static	1	Metrica int	20 (E2)	0	0
RIP	-	infinite	20 (E2)	0	Metrică IGP
EIGRP	infinite	-	20 (E2)	0	Metrică IGP
OSPF	Infinite	Infinite	20 (E2)	0	Metrică IGP
ISIS	Infinite	Infinite	20 (E2)	0	Metrică IGP
BGP	infinite	infinite	1 (E2)	0	Metrică IGP

Figura 7-2: Metrici de redistribuire

7.1.2 Problema rutării suboptimale

Se va considera un scenariu de fuziune a două companii. Compania A foloseşte RIPv2 ca şi protocol de rutare iar compania B, OSPF. Din motive de eficienţă a direcţionării fluxurilor de trafic prin reţea, s-a stabilit ca redistribuţia între cele două reţele să se facă în cel puţin două puncte. De asemenea, ambele companii doresc toate informaţiile de rutare prin ambele puncte astfel încât administratorul a decis să implementeze redistribuţie bidirecţională.

Se va presupune reţeaua 10.10.10.0/24 direct conectată la unul din ruterele companiei A. Administratorul va configura mai întâi redistribuţia pe ruterul R2 iar apoi pe R4. Astfel reţeaua 10.10.10.0 va fi transmisă prin ruterul R2 în domeniul OSPF. Ruterul R3 va trimite imediat mai departe această reţea către R4 căci OSPF foloseşte mesaje de *update* declanşate de evenimente. În acest moment, R4 are deja această reţea în tabela de rutare cunoscută prin RIP cu o distanţă administrativă de 120. Aceeaşi reţea a venit însă din partea OSPF cu o distanţă administrativă de 110. Astfel ruterul va lua decizia de a instala în tabela de rutare ruta ce trece prin compania B, ignorând calea mai scurtă, direct prin ruterul R1, deoarece distanţa administrativă a protocolului OSPF este mai bună decât a lui RIP. Orice trafic către reţeaua 10.10.10.0/24 ce va ajunge de acum la R4 va fi rutat suboptimal prin ruterul R3.

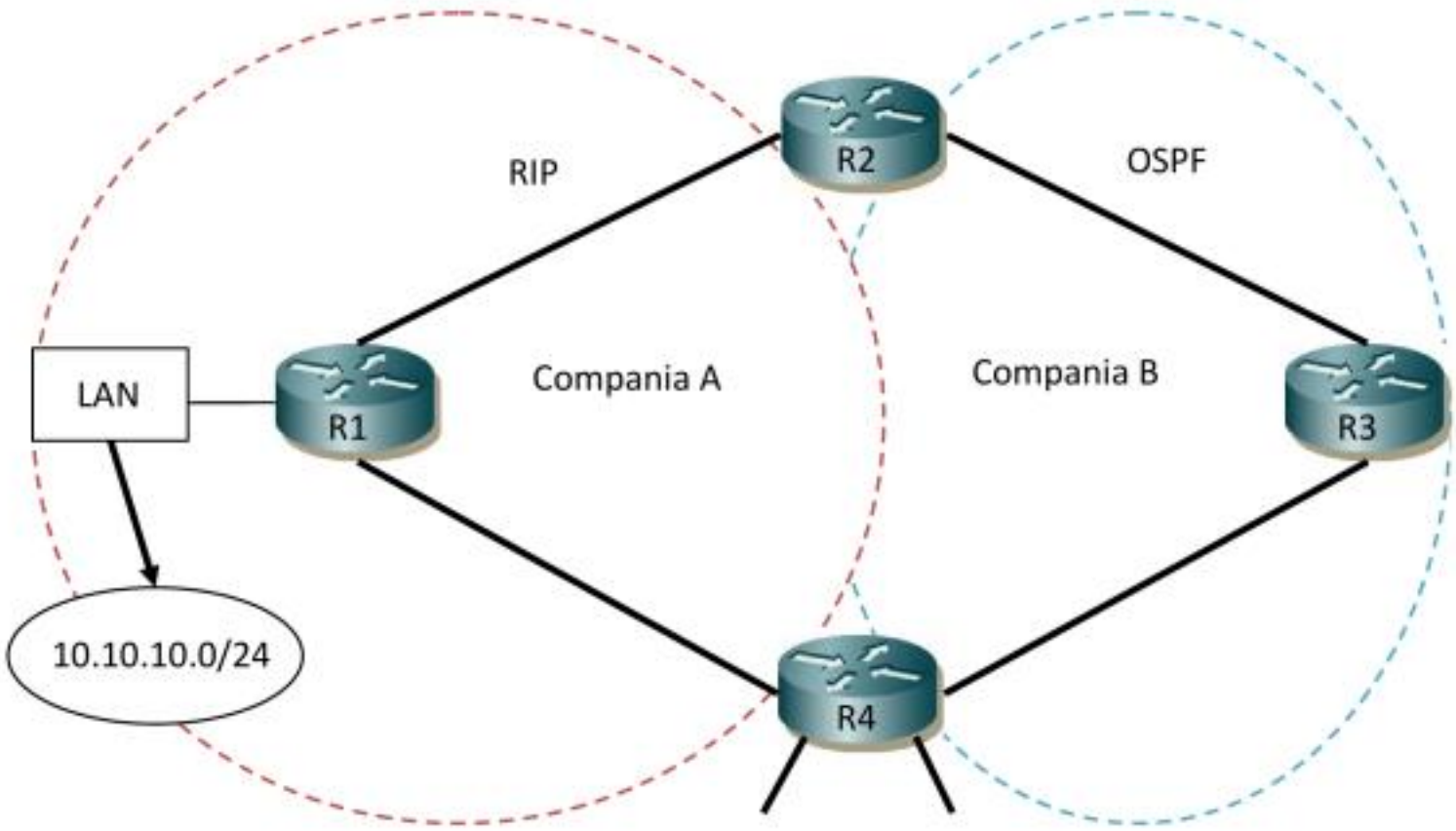

Figura 7-3: Rutare suboptimală

Acest fenomen poartă numele de *route feedback* şi apare întotdeauna când o rută trece dintr-un protocol cu distanţa administrativă mare într-un protocol cu distanţa administrativă mai mică şi apoi înapoi în protocolul iniţial prin alt punct de redistribuţie în reţea.

Se ridică totuşi unele întrebări în continuarea scenariului de mai sus:

1. De ce fenomenul de *route-feedback* se produce la R4 şi nu la R2 ?
2. R4 este şi el punct de redistribuţie bidirecţională. De ce nu redistribuie ruta 10.10.10.0 din RIP în OSPF, urmând ca şi R2 să aibă *route-feedback?*
3. R4 va face rutare suboptimală doar pentru reţeaua 10.10.10.0/24 ?

1. Motivul pentru care R4 ajunge să facă rutarea suboptimală, şi nu R2, este pentru că cel din urmă a fost primul ruter pe care s-a activat redistribuţia. Orice protocol de rutare din prezent, inclusiv RIPv2, are facilitatea transmiterii *update*-urilor declanşate de evenimente; astfel reţeaua 10.10.10.0/24 a ajuns printr-un mesaj de *update* OSPF mult mai repede decât a reuşit R4 să iniţializeze procesul de redistribuţie.

2. Chiar dacă R4 este punct de redistribuţie bidirecţională, odată ce acesta a instalat în tabela de rutare o rută OSPF către 10.10.10.0/24, nu mai poate redistribui această rută din RIP în OSPF pentru că nu mai cunoaşte această rută prin primul din cele două protocoale. Este aplicată regula definită astfel: *pentru a putea redistribui o rută din protocolul X în protocolul Y, acea rută trebuie să fie prezentă în tabela de rutare ca fiind cunoscută prin protocolul X.*

3. Nu doar reţeaua 10.10.10.0 va fi afectată de *route feedback*. Atât reţeaua dintre R1 si R2 cât şi oricare altă reţea din compania A ce nu este direct conectată la R4 va fi susceptibilă la această problemă.

Pentru evaluarea exemplului de mai sus este foarte important conceptul cheie de distanţă administrativă. Definim astfel problema:

Faptul ca ruterul R4 nu a putut face diferenţa între rute interne domeniului OSPF si rute ce au fost redistribuite în OSPF a produs rutarea suboptimală.

După cum s-a precizat în observațiile făcute la Figura 5-1, există protocoale care pot face în mod implicit diferența între rute externe şi interne prin intermediul distanței administrative.

Se va presupune acelaşi scenariu de mai sus, înlocuind însă protocolul OSPF cu EIGRP. Când ruta către rețeaua 10.10.10.0/24 va fi introdusă în EIGRP prin ruterul R2, distanța sa administrativă va fi setată la valoarea 170, implicită pentru rutele externe EIGRP. Ruterul R4 va alege acum între o rută cu distanța 170 (EIGRP) şi o altă rută cu distanța 120 (RIP). Este evident faptul că fenomenul de *route feedback* nu va mai apărea, căci R4 va alege calea corectă, prin RIP.

Deşi EIGRP rezolvă problema rutării suboptimale definite mai sus, această soluție nu este aplicabilă în toate cazurile.

Pentru a obține o definiție completă a problemei se va extinde scenariul adăugând un domeniu OSPF adiacent domeniului EIGRP.

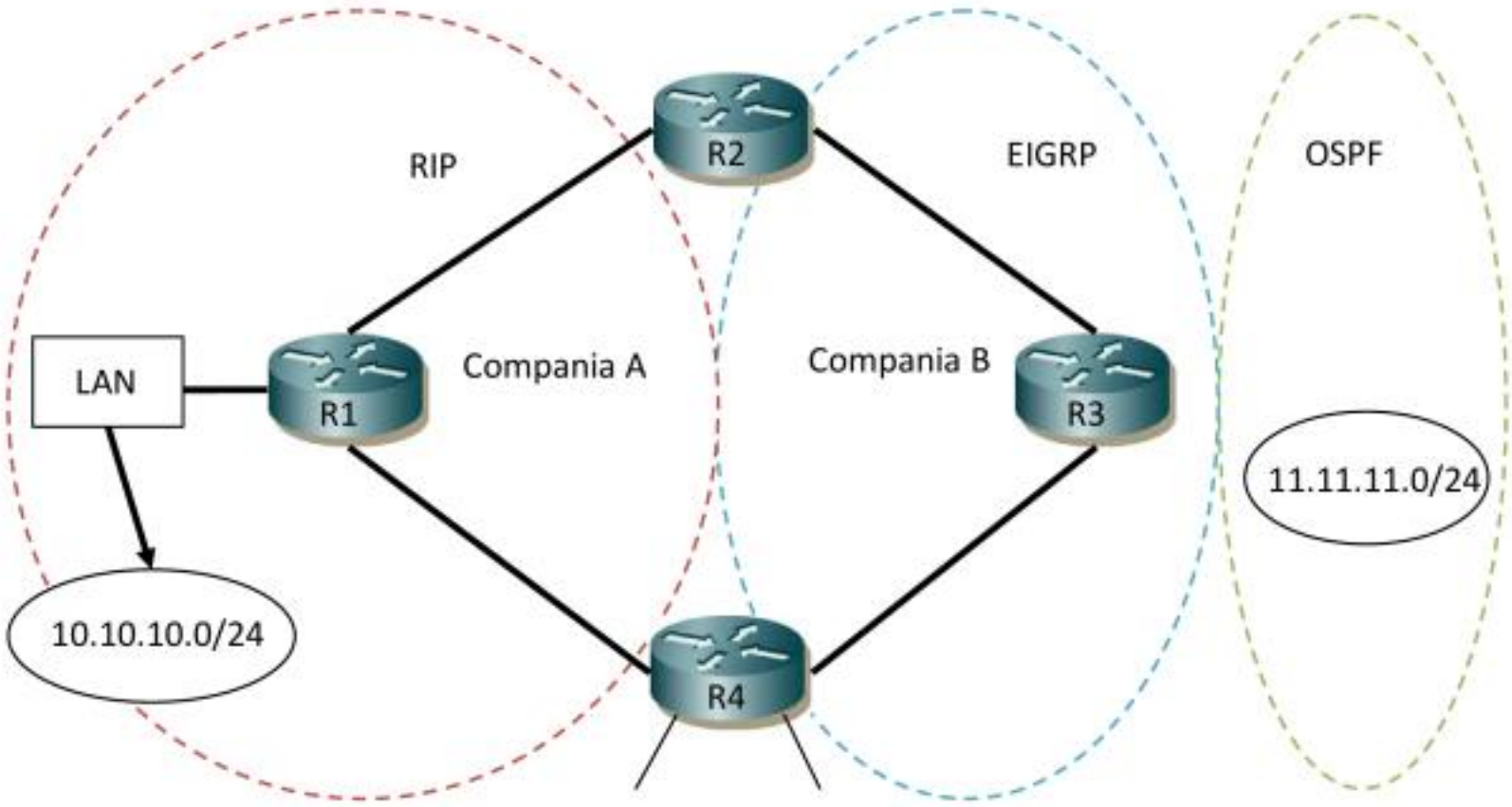

Figura 7-4: Rutare suboptimală 2

Ruterul R3 va fi unicul punct de redistribuție bidirecțională dintre EIGRP şi OSPF. O rețea oarecare 11.11.11.0/24 va fi introdusă astfel în EIGRP şi va primi o distanță administrativă de 170, ca orice altă rută externă. Ruta va fi apoi redistribuită prin R2 sau R4 (depinde care dintre cele două este mai rapid) în compania A. Se va presupune că ruterul R2 este mai rapid şi ca a fost primul ce a introdus 11.11.11.0/24 în RIP. Ruta va fi deci transmisă prin domeniul RIP până la ruterul R4.

Ca şi în cazul anterior, ruterul R4 va avea deja o rută către 11.11.11.0/24 cu distanța administrativă 170 prin domeniul EIGRP. Ruta venită de la R1 prin RIP va avea însă distanța administrativă 120 deci R4 o va prefera pe aceasta. Astfel se va produce din nou rutare suboptimală pentru orice trafic ce ajunge la ruterul R4 fiind destinat rețelei 11.11.11.0/24. Devine astfel evidentă nevoia unui mecanism de identificarea uşoară a unor rețele redistribuite şi de manipularea acestora.

În capitolele ce urmează se vor prezenta metodele cele mai folosite în rezolvarea problemei descrise mai sus.

7.1.3 Bucle de rutare

Buclele de rutare sunt un caz particular şi mult mai grav al rutării suboptimale. Acestea pot apărea în rețele ce conțin cel puțin trei protocoale diferite de rutare şi se rezolva adesea cu destul de mult efort administrativ. Apariția buclelor de rutare nu se va discuta însă în acest capitol deoarece conceptul depăşeşte scopul acestei cărți. Mai multe informații legate de acest subiect pot fi obținute din cărți de specialitate[1].

7.2 Manipularea rutării

7.2.1 Manipularea *update*-urilor de rutare

Mesajele pe care orice protocol de rutare îl are în comun este *update*-ul de rutare. Fie că se numeşte LSA sau LSP, că se trimite la un interval fix de timp sau că este declanşat de o schimbare în rețea, acesta este folosit de orice protocol pentru a transmite adrese de rețea şi metricile ataşate acelor adrese. Prin manipularea mesajelor de *update* se influențează cantitatea de informație pe care un ruter o are la dispoziție pentru a lua o decizie de direcționare, rezolvându-se implicit şi problemele de rutare suboptimală sau buclele de rutare.

Cisco IOS conține mai multe mecanisme prin care se pot controla rutele atât în procesul de redistribuție cât şi în afara acestuia. În acest sens, capitolele ce urmează vor prezenta:

- Comanda *passive-interface* – pasivizează o interfață pentru un protocol de rutare.În funcție de tipul protocolului de rutare, efectele comenzii sunt diferite.
- Comanda *distance* – **poate schimba distanța administrativă pentru anumite rute învățate dintr-un protocol specific**
- **Mecanismul** *distribute-list* **– principalul mecanism de filtrare de rute**
- **Mecanismul** *route-map* **– se foloseşte atât în cele mai avansate tehnici de prevenire a rutării suboptimale (***route tagging***) cât şi în** *traffic engineering şi în protocolul BGP*

7.2.1.1 Pasivizarea interfețelor

Cea mai puțin complexă şi în acelaşi timp cea mai radicală metodă de manipulare a *update*-urilor de rutare este pasivizarea unei interfețe. În funcție de protocolul de rutare folosit, o interfață pasivizată se poate comporta în diferite moduri. Pasivizarea unei interfețe se face în Cisco IOS prin comanda **`passive-interface`** la nivel de interfață.

Protocoale *distance-vector*

Un protocol *distance-vector* va înceta să trimită *update*-uri de rutare pe o interfață pasivizată, însă va continua să le primească. Acest lucru este foarte important în protocoale precum RIP deoarece în acesta comanda **`network`** nu poate activa protocolul decât pe rețele *classful*.

Pentru a exemplifica acest comportament se va folosi topologia de mai jos.

În figura 7-5 ruterul R1 este direct conectat la două rutere de nivel distribuție şi la o rețea locală. Fiind o rețea de dimensiune mică, pentru comunicarea de rute între cele 3 rutere se foloseşte protocolul RIP. La introducerea comenzii **`network`**, procesul RIP va fi activat pe toate interfețele ruterului R1 deoarece comanda **`network`** se activează pe rețele classful. Astfel, pe interfața Fa0/0 vor fi trimise mesaje de *update* ce nu vor ajunge pe niciun alt ruter. Mai mult, aceste mesaje pot reprezenta un risc de securitate căci comunică oricărei stații de pe rețeaua locală rute cunoscute prin RIP.

---

[1] Cisco.Press.CCIE.Routing.and.Switching.Exam.Certification.Guide.3rd.Edition.Nov.2007

Soluția pentru această problemă este pasivizarea interfeței Fa0/0, fapt ce va duce la suprimarea *update*-urilor trimise pe acea interfață.

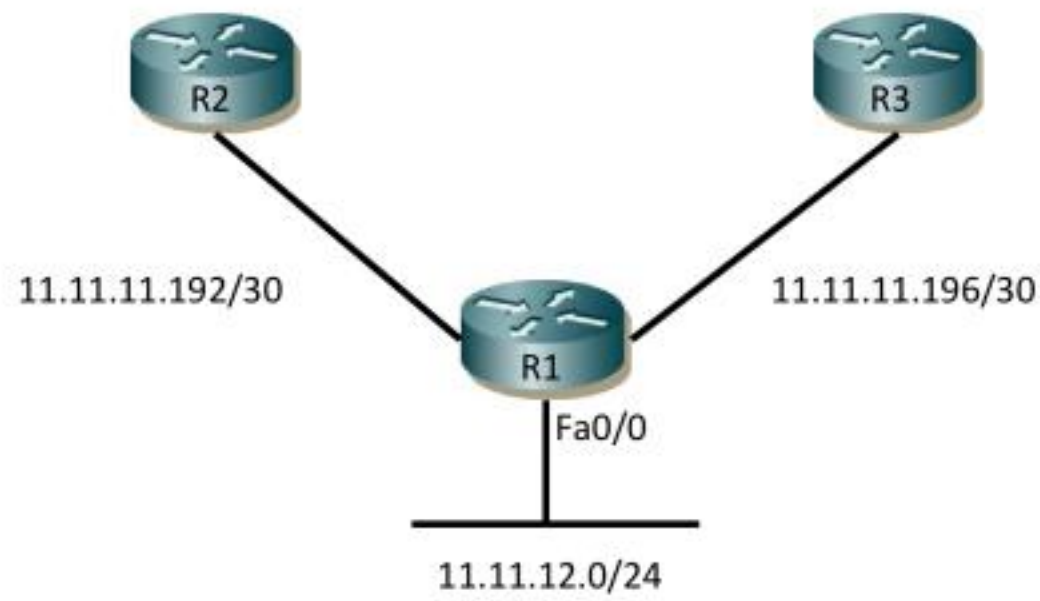

Figura 7-5: *passive-interface* RIP

Protocoale *link-state*

În cazul *link-state*, pasivizarea unei interfețe duce atât la blocarea mesajelor de *update* direcționate spre ruter, cât şi la blocarea mesajelor de *hello*. Activarea comenzii **`passive-interface`** între două rutere va duce astfel la pierderea adiacenței între acestea. Utilitatea comenzii este aceeaşi ca şi în cazul *distance-vector,* însă cu un scenariu puțin diferit. Deşi orice protocol *link-state* permite specificarea unui *wildcard mask* pentru identificarea rețelei exacte pe care se doreşte activarea protocolului, câteodată legăturile către un ruter sau switch de nivel 3 sunt prea numeroase pentru a se introduce comanda **`network`** pentru fiecare rețea în parte. În astfel de situații, de multe ori se preferă activarea la nivel de *supernet* şi dezactivarea câtorva interfețe pe care nu se doreşte rularea protocolului.

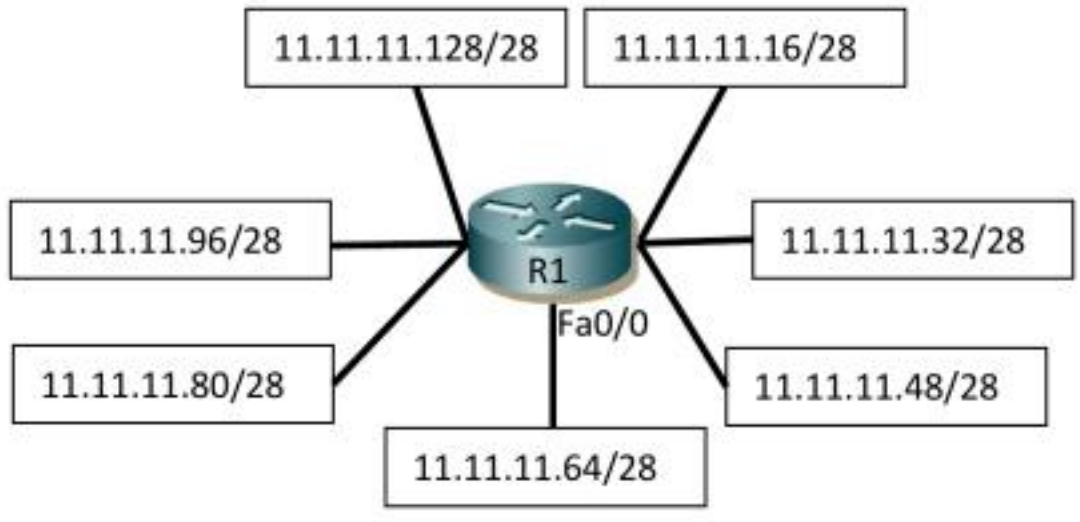

Figura 7-6: Passive-interface link-state

În figura de mai sus, pentru a activa OSPF pe toate interfețele în afară de cea dinspre rețeaua 11.11.11.64/28, se foloseşte comanda **`network 11.11.11.0 0.0.0.255 [area 0]`**, după care

se pasivizează interfața Fa0/0. Acest mod de operare este mult mai rapid decât folosirea a şapte comenzi **network**.

7.2.1.2 Schimbarea distanței administrative

La orice redistribuție de rute intervin pierderi de informație în distanța administrativă şi metrica unei rețele. Se va reaminti topologia dată ca exemplu în capitolul 5.1.3:

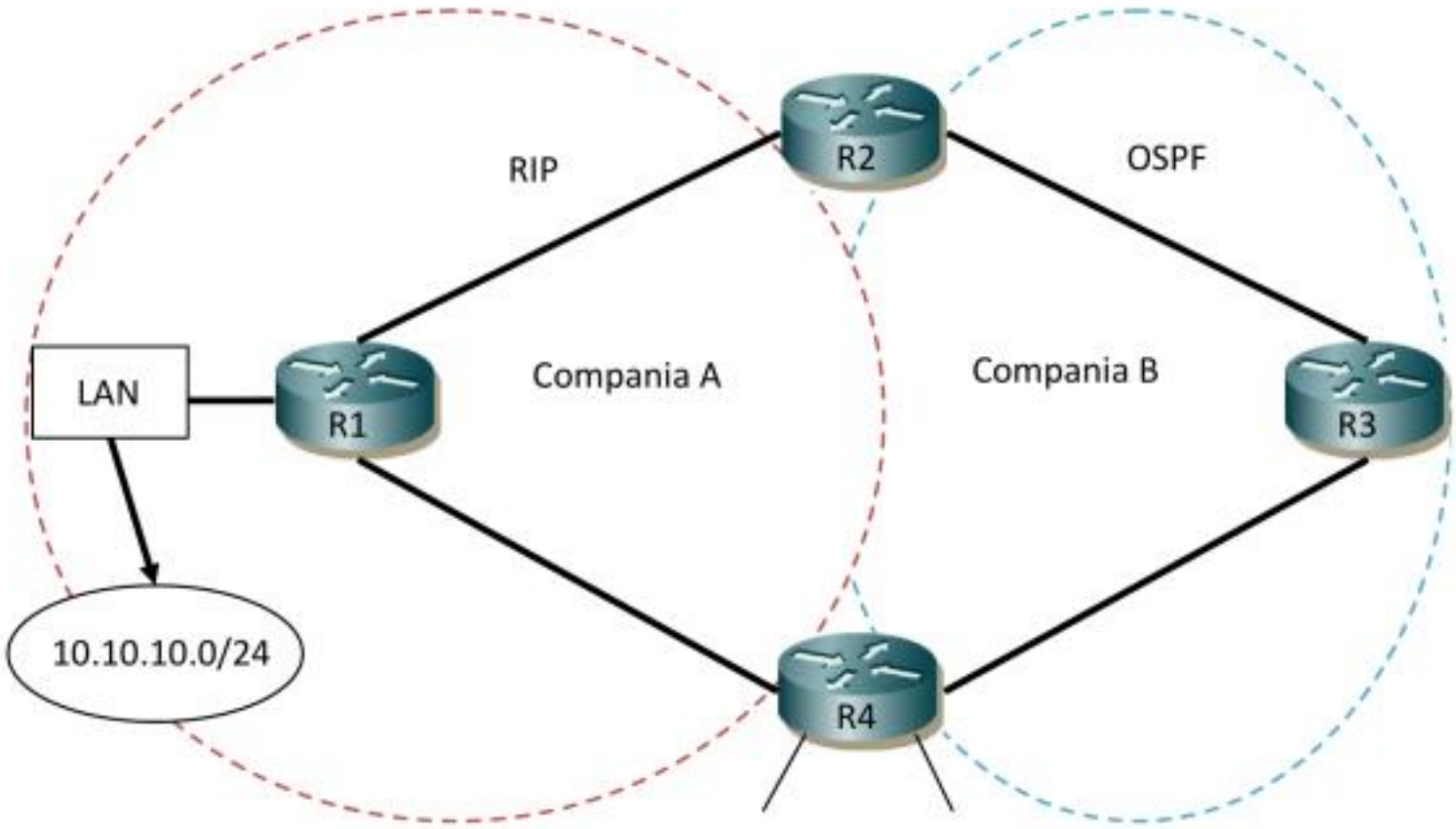

Figura 7-7: Rutare suboptimală

Ca şi în cazul anterior, se va presupune R2 ca fiind primul punct în care se realizează redistribuție bidirecțională. La intrarea în spațiul OSPF, ruta 10.10.10.0 îşi pierde distanța administrativă de 120 şi metrica de 1 hop, urmând ca acestea să fie setate la valorile implicite oferite de OSPF. Ruterul R4 va ajunge astfel să efectueze o comparație între o rută RIP către 10.10.10.0 cu distanța 120 şi o rută OSPF către aceeaşi rețea cu o distanța 110. Alegerea incorectă a drumului se face datorită distanței administrative. Aici intervine mecanismul de schimbare a distanței administrative folosind comanda *distance*. Aceasta oferă două funcționalități:

- Identificarea unor rețele afectate de comanda **distance** , folosind un ACL
- Modificarea distanței administrative pentru rețele identificate pentru a preveni o decizie suboptimală de rutare

Prin comanda **distance**, în procesul OSPF de pe ruterul R4 se vor identifica toate rețelele RIP ce nu sunt direct conectate la acest ruter şi li se va schimba distanța administrativă la o valoare mai mică de 110 înainte de introducerea lor în tabela de rutare. Cu această modificare, R4 va alege întotdeauna calea corectă indicată de noua distanță administrativă configurată. Identificarea rețelelor pentru care distanța administrativă va fi schimbată în rutele asociate se face folosind o listă de acces. Deşi până la capitolul acesta *ACL*-urile au fost folosite doar pentru a bloca sau permite trafic, acestea sunt, de fapt, mecanismul general folosit în IOS pentru identificarea rutelor şi destinațiilor.

Deoarece la redistribuția bidirecțională nu se poate determina care dintre cele două rutere de *border* va inița primul procesul de redistribuție, mecanismul de schimbare a distanței administrative trebuie aplicat pe ambele rutere.

Sintaxa şi aplicarea comenzii **distance** va fi studiată în capitolul 5.3.

7.2.1.3 Filtrarea rutelor

Față de mecanismul de schimbare a distanței administrative, filtrare de rute nu presupune modificarea metadatelor unei rute ci pur şi simplu împiedicarea învățării rutei de către protocol. În Cisco IOS acest scop este atins folosind **distribute-list**.

Aplicând conceptul în topologia anterioară, R4 va folosi un *distribute-list* pentru a identifica rețele provenite din RIP pe care le primeşte de la R3 şi apoi pentru a le filtra din tabela de rutare. Astfel R4 nu va ajunge niciodată la procesul de alegere între o rută RIP şi alta OSPF către aceeaşi rețea, căci va cunoaşte întotdeauna doar rute RIP către rețelele din domeniul RIP şi doar rute OSPF către rețelele din domeniul OSPF.

Identificarea rețelelor ce trebuie filtrate într-un *distribute-list* se face ca şi în comanda *distance*, folosind un ACL. Mai multe detalii de sintaxă vor fi oferite în capitolul 5.3

Comportamentul de filtrare a unui *distribute-list* este însă diferit pentru protocoale *distance-vector* şi *link-state*.

În protocoalele *distance-vector* un *distribute-list* va filtra complet rutele identificate astfel încât acestea nu se vor găsi niciunde în memoria ruterului. Protocoalele link state păstrează rutele primite în tabela de topologie, însă cât timp *distribute-list*-ul este activ, nu le promovează niciodată în tabela de rutare.

7.2.2 Policy-based routing

Rutarea IGP[1] are ca principal scop găsirea celei mai bune rute bazate pe o metrică relevantă, către o destinație posibilă. Câteodată sunt necesare unele excepții de la această regula, excepții generate de anumite decizii de politică. Procesul prin care o decizie a protocolului de rutare este suprascrisă de o politică ce a fost configurată manual poartă numele de rutare bazată pe politică sau **Policy-based routing** (PBR).

7.2.2.1 Route-maps

În Cisco IOS, *Policy-based routing* este implementat folosind *route-map*-uri. Un *route-map* are o logică asemănătoare cu structurile if/then/else folosite în limbajele de programare.

Pentru a identifica diferite tipuri de trafic, un *route-map* foloseşte comenzi **match** care pot folosi în spate liste de acces sau diferite criterii de rutare (ex: interfața de intrare a pachetelor). După ce anumite pachete au fost identificate, *route-map*-ul poate specifica şi anumite acțiuni pentru acele pachete folosind comenzi **set**. Acest aspect este cel care face din *route-map* un instrument foarte puternic. *Route-map*-urile sunt folosite în realizarea mai multor configurații:

- PBR
- Manipularea atributelor BGP
- Controlul redistribuirii

Figura de mai jos prezintă structura unui *route-map*.

[1] http://en.wikipedia.org/wiki/Interior_gateway_protocol

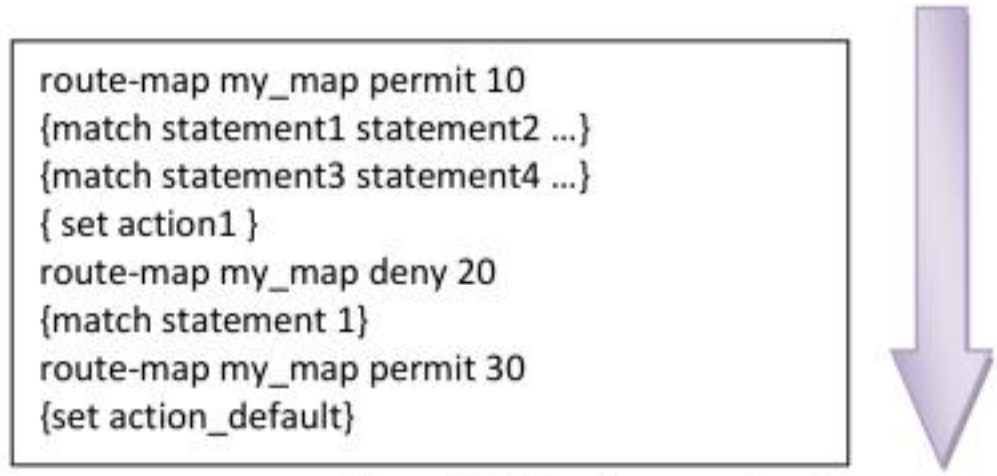

Figura 7-8: Structura unui route-map

Un *route-map* este construit din mai multe reguli ce sunt parcurse secvențial, precum regulile unui ACL. Fiecare regulă este precedată de cuvântul cheie *route-map*, numele *route-map*-ului, acțiunea globală (*permit/deny*) şi un index relevant în ordinea în care se procesează intrările din *route-map.*

După cum se poate observa, intrările de mai sus conțin indecşi din zece în zece unități. Acest mod de a scrie un *route-map* este un *best practice* ce permite inserarea facilă a altor intrări intermediare în cazul în care se doreşte modificarea logicii *route-map*-ului. Pentru ca sistemul de operare să poată lega toate regulile *route-map*-ului într-o listă, va trebuie ca fiecare regulă să aibă acelaşi nume cu cel al *route-map*-ului (în cazul de mai sus: *my_map*).

Fiecare regulă are una sau mai multe comenzi *match* care reprezintă modul de identificare a traficului. Fiecare comandă *match* poate avea mai multe argumente. Se prezintă astfel două cazuri de evaluare logică a *statement*-urilor:

- Dacă există mai multe *statement*-uri în cadrul unei singure comenzi *match*, se va aplica ŞI LOGIC între acestea. Cu alte cuvinte trebuie ca toate aserțiunile despre respectivul trafic să fie adevărate pentru a face o potrivire cu acea regulă.
- Dacă există mai multe comenzi *match* în cadrul unei reguli, se va aplica SAU LOGIC între acestea. Cu alte cuvinte trebuie ca cel puțin o comandă *match* să fie evaluată ca fiind adevărată pentru a face o potrivire cu acea regulă.

După ce s-a realizat o potrivire cu o anumită regulă, procesorul va încerca să execute acțiunea precizată de comanda *set*. Prin aceasta se pot schimba atribute ale rutei sau se poate suprascrie logica de rutare a protocolului de rutare (PBR). Dacă nu există nicio regulă *set* sau toate regulile *set* prezente au fost aplicate, se va aplica **acțiunea globală**. Aceasta poate fi *permit* sau *deny*, însă în funcție de mecanismul ce foloseşte *route-map*-ul cele două cuvinte cheie au diferite interpretări.

Ca şi exemplu, dacă mecanismul este PBR, o acțiune globală *permit* va însemna că acest trafic este introdus în PBR şi este rutat în funcție de politica aplicată în comanda *set*. Dacă acțiunea globală este *deny*, înseamnă ca acest tip de trafic nu trebuie să fie rutat în funcție de politică şi trebuie introdus în procesul normal de rutare.

În contrast, dacă mecanismul este controlul redistribuirii, o acțiune globală *deny* va însemna că respectiva ruta nu va fi redistribuită. Mai multe detalii despre acest proces vor fi oferite în următorul capitol.

Ambele directive (*set* şi *match*) pot lipsi dintr-o regulă de *route-map*. Dacă directiva *set* lipseşte înseamnă că administratorul nu doreşte alterarea acelui trafic în niciun mod. Lipsa directivei *match* este echivalentă cu un statement de tipul *match any*. Cu alte cuvinte, orice trafic va fi potrivit cu acea regulă.

Ca şi un ACL, orice *route-map* are la sfârşitul listei de reguli o regulă invizibilă care identifică tot traficul şi care are acțiunea globală *deny*. Acesta este motivul pentru care în *route-map*-ul

exemplificat mai sus ultima regulă nu conține directiva *match* (pentru a face potrivire cu orice fel de trafic) şi are ca acțiune globală *permit*. Acest mod de configurare depinde însă de logica *route-map*-ului. Dacă acesta a fost construit pentru a permite doar câteva pachete specifice, regula logică invizibilă *match none* de la sfârşit va fi potrivită. Dacă se doreşte însă potrivirea pe toate pachetele în afară de câteva rețele, se va adăuga o regulă ca ultima din exemplu.

7.3 Configurări în optimizarea rutării

7.3.1 Configurarea redistribuirii între protocoale de rutare

Există o singură comandă prin care se realizează redistribuția şi anume comanda **`redistribute`**. Această comandă este implementată în modul **`(config-router)#`** şi, în funcție de politicile de rutare, poate primi o serie de parametri ce pot influența metrici, tipuri de rute redistribuite sau rețele specifice pentru a fi redistribuite.

Partea uşor confuză în efectuarea configurărilor de redistribuție este realizarea diferenței între protocolul în care se face redistribuția şi protocolul din care se face redistribuția. Cel mai simplu mod de a reține este prin următoarea regulă:

Protocolul ***în care*** *se realizează redistribuția este protocolul dat ca parametru comenzii* **`router`**, *iar protocolul* ***din care*** *se realizează redistribuția este protocolul dat ca parametru comenzii* **`redistribute`**.

```
Waters(config)# router ospf 1
Waters(config-ruter)#redistribute rip
% Only classful networks will be redistributed
```

Urmărind regula de mai sus, pentru că se dorea redistribuția în OSPF s-a folosit comanda **`ruter ospf 1`**, iar pentru că protocolul redistribuit era *RIP*, acesta a fost dat ca parametru comenzii **`redistribute`**. Se poate observa că la comanda de redistribuție sistemul de operare a returnatun mesaj de avertizare. Dată în această formă, comanda va introduce în OSPF doar rețele classful (A, B sau C). Pentru a rezolva această problemă trebuie adăugat cuvântul cheie **`subnets`**.

De asemenea, comanda **`redistribute`** nu a precizat mai sus nicio metrică de intrare în protocol (*seed metric*). Pe lângă ISIS, doar în OSPF se pot redistribui rute din alt protocol fără a preciza manual metrica de intrare. Orice rută externă adusă în OSPF va avea o metrică implicită de 20 şi un *metric-type* E2. Este recomandat ca în topologii şi scenarii reale să fie modificați aceşti parametrii pentru a asigura o diferențiere între multiplele rute de ieşire din arie (dacă acestea există) folosind parametrii: **metric** şi **metric-type**.

EIGRP foloseşte o metrică construită din maxim 5 variabile: *bandwidth*, *delay*, *reliability*, *load*, *MTU*. În mod implicit sunt folosite doar *bandwidth* şi *delay* cu valorile configurate pe fiecare interfață. Calculul metricii se realizează după formula EIGRP:

EIGRP Metric = $256*([K_1*Bw + K_2*Bw/(256-Load) + K_3*Delay]*[K_5/(Reliability + K_4)])$

Datorită constantelor „K" setate la 0 pentru valorile *load*, *reliability* sau *delay*, acestea nu vor participa în mod implicit în formulă.

În general EIGRP poate fi configurat pentru a fi destul de exact folosind doar metricile implicite. Când se redistribuie orice protocol de rutare în EIGRP trebuie specificate toate variabilele ce formează metrica, chiar dacă vor fi folosite toate.

```
Waters(config)# router eigrp 100
Waters(config-ruter)#redistribute ospf 1 metric 10000 100 255 1 1500
```

La redistribuția în EIGRP doar rutele direct conectate și cele statice sunt introduse cu o metrică de intrare validă. Valoarea acesteia este calculată cu parametrii *bandwidth* și *delay* ai interfeței direct conectate, respectiv interfeței de ieșire a rutei statice.

ISIS are un comportament prin care setează metrica oricărei rute externe la valoarea „0". Acest lucru nu înseamnă ca refuză să o redistribuie deoarece o metrică de „0" este validă în ISIS. La redistribuția ISIS în alt protocol de rutare se poate opta pentru introducerea rutelor level-1, level-2, level-1-2.

```
R1(config-ruter)#redistribute isis ?
  WORD        ISO routing area tag
  level-1     IS-IS level-1 routes only
  level-1-2   IS-IS level-1 and level-2 routes
  level-2     IS-IS level-2 routes only
  ...
```

Până în acest moment s-au discutat comenzile de care este nevoie pentru a obține o redistribuție funcțională între oricare două protocoale de rutare. Există însă situații în care, deși e nevoie de redistribuție, nu se dorește transferul tuturor rutelor dintr-un protocol în altul din motive administrative sau politice. Folosind conceptele de *route-map* discutate în capitolul anterior se pot controla rutele ce vor fi redistribuite. Comanda **redistribute** poate primi sub orice protocol de rutare parametrul **route-map** prin care se pot identifica rețelele specifice ce se doresc a fi redistribuite. Astfel se produce un fel de „filtrare" a rutelor prin evitarea redistribuirii unor rețele.

Se va analiza comportamentul unei astfel de tehnici de filtrare în configurarea redistribuției protocolului RIP în OSPF.

```
router ospf 1
redistribute rip subnets route-map RIP_TO_OSPF
!
ip access-list standard RIP_TO_OSPF
 permit 10.1.1.0 0.0.0.255
 permit 192.4.5.0 0.0.0.255
 permit 141.99.56.0 0.0.0.255
!
route-map RIP_TO_OSPF permit 10
 match ip address RIP_TO_OSPF
 set metric +10
```

Deoarece redistribuția se face în OSPF, acesta este protocolul dat ca și parametru comenzii **router**. Comanda **redistribute** este dată cu parametrul **subnets** alături de *route-map*-ul cu numele RIP_TO_OSPF. Mai jos este definit acest *route-map* care face potrivire pe regula 10 cu o listă de acces ce poartă același nume. Acțiunea pentru această intrarea este incrementarea metricii de intrare în OSPF cu valoarea 10 (deci în mod implicit intrarea în OSPF se va face cu metrica 30). Acțiunea globală a intrării din *route-map* este *permit*. Prin acest cuvânt cheie se specifică realizarea redistribuției dacă și numai dacă rețelele ce se doresc a fi redistribuite se regăsesc în lista de acces RIP_TO_OSPF. Dacă rețeaua ce se dorește redistribuită nu poate fi identificată de lista de acces de mai sus, ruterul va procesa următoarea intrare din *route-map* care este de fapt intrarea implicită invizibilă ce aplică acțiunea *deny* pe orice rețea. Astfel vor fi redistribuite doar rețelele pe care se face potrivirea în intrarea 10 a *route-map*-ului.

7.3.2 Configurarea filtrării rutelor

Una dintre cele mai cunoscute și utilizate metode de filtrare a rutelor este mecanismul *distribute-list*. Acest tip special de listă de acces este folosit pentru a identifica rutele din *update*-urile de rutare ale protocoalelor și pentru a aplica pachetelor ce transportă aceste *update*-uri una dintre cele două acțiuni implicite ale unei liste de acces uzuale: *permit* sau *deny*. În comparație cu o listă de acces simplă, un *distribute-list* este folosit pentru a filtra rute în locul pachetelor de date.

Un *distribute-list* poate fi aplicat într-un dintre cele două modalități:

- **Filtru de intrare**: în acest caz *distribute-list*-ul poate specifica doar interfața de intrare pe care să fie filtrate *update*-urile. În cazul în care această interfață nu este specificată pentru un filtru de intrare, acesta se va aplica pe toate interfețele ruterului.
- **Filtru de ieșire**: în acest caz *distribute-list*-ul poate specifica atât interfața de ieșire cât și protocolul de rutare în care nu trebuie trimise rutele specificate.

Pentru a observa comportamentul unui *distribute-list* se vor analiza următoarele configurări care previn trimiterea *update-urilor* ce conțin rețelele 10.10.1.0/24 și 192.168.1.0/24 în cadrul protocolului RIPv2.

```
router rip
 network 18.0.0.0
 distribute-list filter_private in Serial1/0
!
ip access-list standard filter_private
 deny   10.10.1.0 0.0.0.255
 deny   192.168.1.0 0.0.0.255
 permit any
!
```

Pentru a identifica rețelele ce trebuie filtrate, comenzii *distribute-list* i-a fost dată ca parametru lista de acces numită *„filter_private"*. Direcția de filtrare a fost specificată ca fiind *„in"*, activă doar pe interfața Serial1/0. Astfel ruterul va filtra rețelele specificate în lista de acces la intrarea *update*-urilor de rutare pe interfața Serial1/0. Este necesară multă atenție la configurarea unui *distribute-list* deoarece, față de un *route-map*, unde potrivirea rutelor se făcea folosind directiva *permit* din ACL, în acest caz specificarea rutelor ce se doresc filtrate se face cu *deny*.

Una din cele mai frecvente greșeli realizate în configurări de filtrare este omiterea directivei *permit any* de la finalul listei de acces. Deoarece un ACL are în mod implicit o directivă *deny any* la sfârșitul ei, administratorul de rețea trebuie să introducă un *permit any* la sfârșitul unui *distribute-list* pentru a permite orice alte *update*-uri decât cele filtrate.

Atenție! Aplicarea unui distribute-list pe direcția de ieșire pe o interfață nu este posibilă decât pentru protocoalele distance-vector și EIGRP.

7.3.3 Configurarea PBR

După cum s-a discutat în capitolul 5.2.2, mecanismul de PBR se configurează în Cisco IOS folosind *route-map*-uri. Așa și în s ituația prezentă în capitolul 5.3.1, prima parte a *route-map*-ului este folosită pentru a face potrivire cu traficul interesant. Această potrivire se face prin intermediul unei liste de acces cu directive *permit*. Partea din *route-map* prin care se face de fapt *Policy-Based Routing* este comanda *set*. Prin intermediul acesteia se poate specifica suprascrierea deciziei de rutare prin forțarea pachetului spre un anumit vecin sau pe o anumita interfață de ieșire. După crearea *route-map*-ului și al listei de acces folosite pentru potrivire, administratorul trebuie să indice interfața pe care *PBR* va fi activat. Pe fiecare interfață a ruterului poate exista o singura politică de PBR.

În exemplul de mai jos, ruterul va trimite toate pachetele cu adresă sursă din rețeaua 10.10.1.0/24 și adresă destinație 141.85.57.1 pe interfața de ieșire Serial1/0 indiferent de decizia pe care protocolul de rutare o ia.

```
interface Serial1/1
 ip address 18.7.15.1 255.255.255.0
 ip policy route-map web_server
!
ip access-list extended web_server
 permit ip 10.10.1.0 0.0.0.255 host 141.85.57.1
!
route-map web_server permit 10
 match ip address web_server
 set interface Serial1/0
!
```

O greșeală des întâlnită în configurarea *PBR* este setarea interfeței de ieșire în cazul în care aceasta folosește tehnologie de tip *Ethernet*. Deși Proxy-ARP este activ în mod implicit pe rutere Cisco, acesta nu va interveni în rezolvarea problemi iar ruterul nu va ști cu ce adresă MAC destinație trebuie să trimită. Se recomandă ca, la rutarea traficului spre o interfață Ethernet, comanda să specifice dirct *next-hop*-ul, şi nu interfața de ieşire.

De asemenea, trebuie reținut că traficul generat local nu este trecut în mod implicit prin filtrul de PBR. Acest comportament este moștenit de la *ACL*-uri unde traficul generat local pe ruter nu este afectat de către ACL. Cisco oferă însă o altă variantă prin care se pot configura politici pentru trafic local. Acest lucru se face din modul global de configurare din IOS prin comanda: **`ip local-policy route-map <route-map-name>`**.

7.4 Studiu de caz

Următorul scenariu este propus pentru a exemplifica modul de aplicare a conceptelor de optimizare a rutării într-o rețea complexă. Topologia, prezentată in figura 5-7, pretinde o serie de zone, în fiecare dintre acestea rulând câte un protocol de rutare diferit. Vor trebui realizate configurările pentru ca rețeaua să conveargă şi pentru a respecta o serie de cerințe privind modul în care se face rutarea.

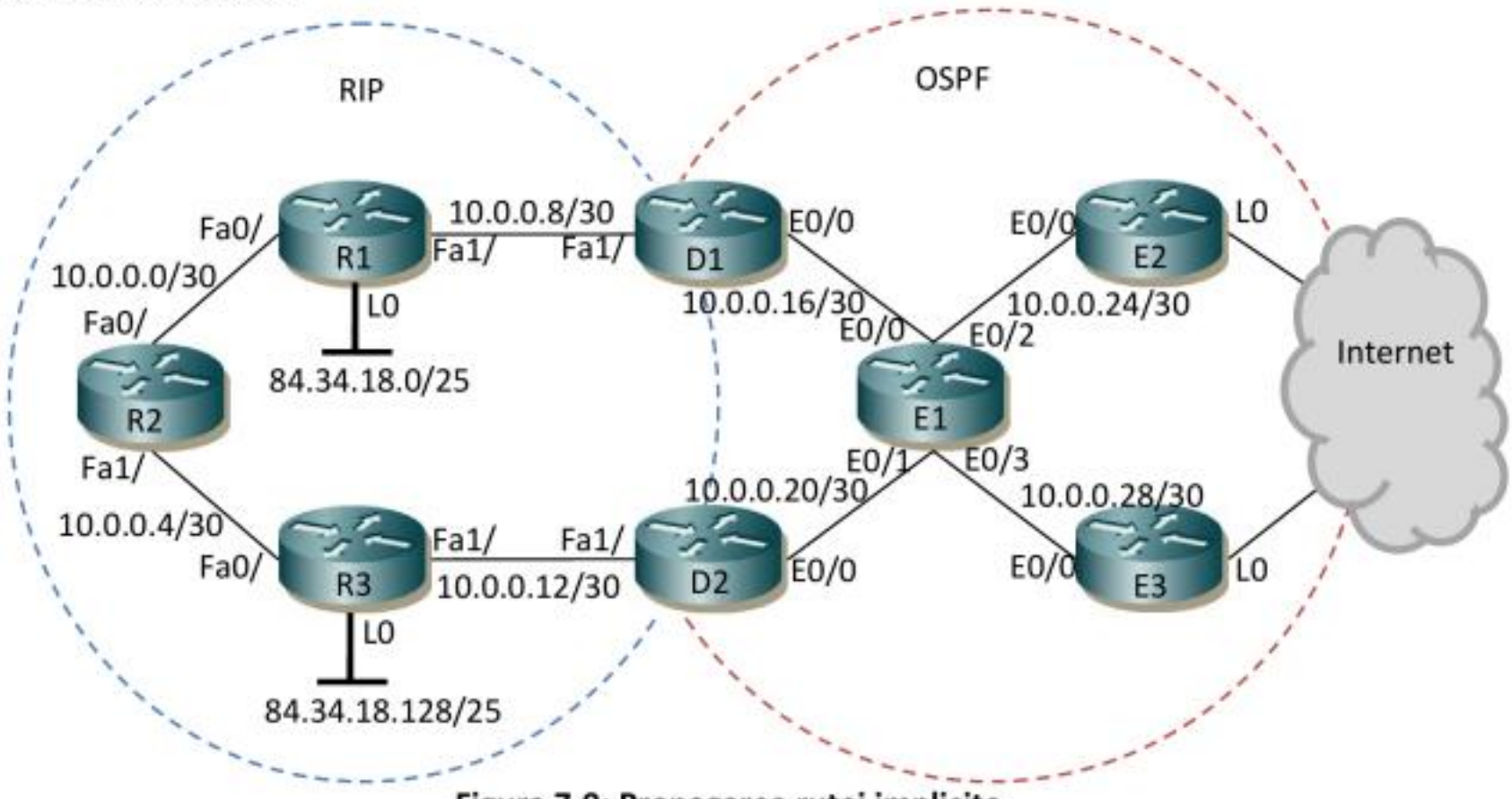

Figura 7-9: Propagarea rutei implicite

7.4.1 Configurări inițiale

Pentru rețeaua de test a fost alocat spațiul de adrese 84.34.18.0/23 pentru rețelele conectate la rutere; pentru legăturile dintre echipamentele de rețea se folosește spațiul de adrese 10.0.0.0/24.

```
R1#show ip interface brief | include up
FastEthernet0/0            10.0.0.1        YES manual up                    up
FastEthernet1/0            10.0.0.9        YES manual up                    up
Loopback0                  84.34.18.1      YES manual up                    up

R2#show ip interface brief | include up
FastEthernet0/0            10.0.0.2        YES manual up                    up
FastEthernet1/0            10.0.0.5        YES manual up                    up

R3#sh ip interface brief | include up
FastEthernet0/0            10.0.0.6        YES manual up                    up
FastEthernet1/0            10.0.0.13       YES manual up                    up
Loopback0                  84.34.18.129    YES manual up                    up

D1#show ip interface brief | include up
Ethernet0/0                10.0.0.17       YES manual up                    up
FastEthernet1/0            10.0.0.10       YES manual up                    up


D2#show ip interface brief | include up
Ethernet0/0                10.0.0.21       YES manual up                    up
FastEthernet1/0            10.0.0.14       YES manual up                    up

E1#show ip interface brief | include up
Ethernet0/0                10.0.0.18       YES manual up                    up
Ethernet0/1                10.0.0.22       YES manual up                    up
Ethernet0/2                10.0.0.25       YES manual up                    up
Ethernet0/3                10.0.0.29       YES manual up                    up

E2#show ip interface brief | include up
Ethernet0/0                10.0.0.26       YES manual up                    up
Loopback0                  84.34.19.1      YES manual up                    up

E3#show ip interface brief | include up
Ethernet0/0                10.0.0.30       YES manual up                    up
Loopback0                  84.34.19.5      YES manual up                    up
```

Interfețele Loopback0 de pe E2 şi E3, ce simulează legătura către Internet, au fost configurate cu adrese din spațiul 84.34.18.0/23.

Ruterele R1, R2 şi R3 sunt slabe ca putere de procesare şi memorie, aşa că s-a luat decizia rulării protocolului RIP în respectiva zonă din rețea. E1, E2 şi E3 fac parte din nucleul rețelei de față, pe aceste rutere fiind necesară rularea protocolul OSPF. E2 şi E3 vor asigura de asemenea şi legătura către Internet a rețelei.

```
R1(config)#router rip
R1(config-router)#no auto-summary
R1(config-router)#version 2
R1(config-router)#network 10.0.0.0
R1(config-router)#network 84.0.0.0

R2(config)#router rip
R2(config-router)#no auto-summary
R2(config-router)#version 2
R2(config-router)#network 10.0.0.0

R3(config)#router rip
R3(config-router)#no auto-summary
R3(config-router)#version 2
R3(config-router)#network 10.0.0.0
```

```
R3(config-router)#network 84.0.0.0

D1(config)#router rip
D1(config-router)#no auto-summary
D1(config-router)#version 2
D1(config-router)#network 10.0.0.0
D1(config-router)#passive-interface ethernet 0/0
D1(config-router)#exit
D1(config)#router ospf 1
D1(config-router)#network 10.0.0.16 0.0.0.3 area 0

D2(config)#router rip
D2(config-router)#no auto-summary
D2(config-router)#version 2
D2(config-router)#network 10.0.0.0
D2(config-router)#passive-interface ethernet 0/0
D2(config-router)#exit
D2(config)#router ospf 1
D2(config-router)#network 10.0.0.20 0.0.0.3 area 0
```

```
E1(config)#router ospf 1
E1(config-router)#network 10.0.0.0 0.255.255.255 area 0

E2(config)#ip route 0.0.0.0 0.0.0.0 loopback 0
E2(config)#router ospf 1
E2(config-router)#network 10.0.0.0 0.255.255.255 area 0
E2(config-router)#network 84.0.0.0 0.255.255.255 area 0
E2(config-router)#default-information originate

E3(config)#ip route 0.0.0.0 0.0.0.0 loopback 0
E3(config)#router ospf 1
E3(config-router)#network 10.0.0.0 0.255.255.255 area 0
E3(config-router)#network 84.0.0.0 0.255.255.255 area 0
E3(config-router)#default-information originate
```

De amintit faptul că, datorită comportamentului *classful* pe care îl are protocolul RIP la activare, se introduc automat în protocol toate rețelele direct conectate ce fac parte, în cazul de față, în 10.0.0.0/8. Pentru ca ruterele D1 şi D2 să nu trimită *update*-uri RIP pe legăturile către E1, au fost pasivizate aceste interfețe din modul de configurare RIP.

Au fost configurate 2 rute *default* pentru rețea, una pe E2 şi una pe E3, ce au fost introdue apoi în OSPF pentru a fi propagate.

În acest moment, în cele două zone din rețea rutele sunt propagate. Tabela de rutare pe D1 (împreună cu D2 sunt singurele ce cunosc atât rute RIP cât şi OSPF) arată astfel:

```
D1#show ip route
Gateway of last resort is 10.0.0.18 to network 0.0.0.0

     84.0.0.0/25 is subnetted, 2 subnets
R       84.34.18.0 [120/1] via 10.0.0.9, 00:00:16, FastEthernet1/0
R       84.34.18.128 [120/3] via 10.0.0.9, 00:00:16, FastEthernet1/0
     10.0.0.0/30 is subnetted, 8 subnets
C       10.0.0.8 is directly connected, FastEthernet1/0
R       10.0.0.12 [120/3] via 10.0.0.9, 00:00:16, FastEthernet1/0
R       10.0.0.0 [120/1] via 10.0.0.9, 00:00:16, FastEthernet1/0
R       10.0.0.4 [120/2] via 10.0.0.9, 00:00:16, FastEthernet1/0
O       10.0.0.24 [110/20] via 10.0.0.18, 00:04:09, Ethernet0/0
O       10.0.0.28 [110/20] via 10.0.0.18, 00:04:09, Ethernet0/0
C       10.0.0.16 is directly connected, Ethernet0/0
O       10.0.0.20 [110/20] via 10.0.0.18, 00:04:09, Ethernet0/0
O*E2 0.0.0.0/0 [110/1] via 10.0.0.18, 00:04:09, Ethernet0/0
```

Pe R1, tabela de rutare arată astfel:

```
R1#sh ip route
Gateway of last resort is not set

     84.0.0.0/25 is subnetted, 2 subnets
C       84.34.18.0 is directly connected, Loopback0
R       84.34.18.128 [120/2] via 10.0.0.2, 00:00:23, FastEthernet0/0
     10.0.0.0/30 is subnetted, 6 subnets
C       10.0.0.8 is directly connected, FastEthernet1/0
R       10.0.0.12 [120/2] via 10.0.0.2, 00:00:23, FastEthernet0/0
C       10.0.0.0 is directly connected, FastEthernet0/0
R       10.0.0.4 [120/1] via 10.0.0.2, 00:00:23, FastEthernet0/0
R       10.0.0.16 [120/1] via 10.0.0.10, 00:00:01, FastEthernet1/0
R       10.0.0.20 [120/3] via 10.0.0.2, 00:00:23, FastEthernet0/0
```

Pe E1, tabela de rutare arată astfel:

```
E1#sh ip route
Gateway of last resort is 10.0.0.30 to network 0.0.0.0

     10.0.0.0/30 is subnetted, 4 subnets
C       10.0.0.24 is directly connected, Ethernet0/2
C       10.0.0.28 is directly connected, Ethernet0/3
C       10.0.0.16 is directly connected, Ethernet0/0
C       10.0.0.20 is directly connected, Ethernet0/1
O*E2 0.0.0.0/0 [110/1] via 10.0.0.30, 00:00:00, Ethernet0/3
               [110/1] via 10.0.0.26, 00:00:00, Ethernet0/2
```

7.4.2 Redistribuirea protocoalelor de rutare

După cum se observă din tabelele de rutare ale celor 3 rutere, cele două zone sunt complet izolate din punct de vedere al rutării. Pentru a rezolva această problemă, trebuie făcută redistribuirea informației de rutare între RIP şi OSPF.

În cazul de față se doreşte realizarea redistribuirii în ambele sensuri, atât pe D1 cât şi pe D2, pentru a folosi la maxim redundanța oferită de topologie. În continuare se prezintă configurațiile ce sunt necesare pentru a realiza acest lucru:

```
D1(config)#router ospf 1
D1(config-router)#redistribute rip subnets
D1(config-router)#exit
D1(config)#router rip
D1(config-router)#redistribute ospf 1 metric 1

D2(config)#router ospf 1
D2(config-router)#redistribute rip subnets
D2(config-router)#exit
D2(config)#router rip
D2(config-router)#redistribute ospf 1 metric 1
```

În momentul în care se redistribuie un protocol in OSPF este important argumentul *subnets*. Dacă acesta nu este precizat, se vor distribui numai clasele majore de IP, nu şi subrețelele acestora.

La redistribuirea în RIP este obligatorie specificarea unei metrici pentru rutele ce vor fi importate. Dacă această metrică nu este specificată, se va presupune metrica 16 (infinit) şi rutele nu vor apărea în tabela de rutare şi nici nu vor fi comunicate altor rutere.

După realizarea redistribuției, tabela de rutare pe R1 şi E1 arată astfel:

```
R1#sh ip route
Gateway of last resort is 10.0.0.10 to network 0.0.0.0

     84.0.0.0/25 is subnetted, 2 subnets
C       84.34.18.0 is directly connected, Loopback0
R       84.34.18.128 [120/2] via 10.0.0.2, 00:00:00, FastEthernet0/0
     10.0.0.0/30 is subnetted, 8 subnets
C       10.0.0.8 is directly connected, FastEthernet1/0
```

```
R       10.0.0.12 [120/1] via 10.0.0.10, 00:00:09, FastEthernet1/0
C       10.0.0.0 is directly connected, FastEthernet0/0
R       10.0.0.4 [120/1] via 10.0.0.2, 00:00:00, FastEthernet0/0
R       10.0.0.24 [120/1] via 10.0.0.10, 00:00:09, FastEthernet1/0
R       10.0.0.28 [120/1] via 10.0.0.10, 00:00:09, FastEthernet1/0
R       10.0.0.16 [120/1] via 10.0.0.10, 00:00:09, FastEthernet1/0
R       10.0.0.20 [120/1] via 10.0.0.10, 00:00:09, FastEthernet1/0
R*   0.0.0.0/0 [120/1] via 10.0.0.10, 00:00:09, FastEthernet1/0

E1#show ip route
Gateway of last resort is 10.0.0.30 to network 0.0.0.0

     84.0.0.0/25 is subnetted, 2 subnets
O E2    84.34.18.0 [110/20] via 10.0.0.17, 00:15:39, Ethernet0/0
O E2    84.34.18.128 [110/20] via 10.0.0.17, 00:15:39, Ethernet0/0
     10.0.0.0/30 is subnetted, 8 subnets
O E2    10.0.0.8 [110/20] via 10.0.0.17, 00:15:39, Ethernet0/0
O E2    10.0.0.12 [110/20] via 10.0.0.21, 00:15:39, Ethernet0/1
O E2    10.0.0.0 [110/20] via 10.0.0.17, 00:15:39, Ethernet0/0
O E2    10.0.0.4 [110/20] via 10.0.0.17, 00:15:39, Ethernet0/0
C       10.0.0.24 is directly connected, Ethernet0/2
C       10.0.0.28 is directly connected, Ethernet0/3
C       10.0.0.16 is directly connected, Ethernet0/0
C       10.0.0.20 is directly connected, Ethernet0/1
O*E2 0.0.0.0/0 [110/1] via 10.0.0.30, 00:15:39, Ethernet0/3
               [110/1] via 10.0.0.26, 00:15:39, Ethernet0/2
```

7.4.3 Probleme la redistribuție

Deşi la o primă privire redistribuirea pare să se fi făcut fără probleme, la o mai atentă analiză a tabelelor de rutare din zona RIP pot fi identificate câteva probleme. De exemplu, ruta către rețeaua 10.0.0.12/30 are ca *next hop* D1, urmând apoi să treacă prin E1 şi D2, deşi ar fi fost mult mai rapid să aibă ca *next hop* pe R2.

Acest comportament poartă numele de rutare suboptimală şi poată să apară în cazul configurării incorecte a redistribuirii. Problema se datorează faptului că redistribuția a fost realizată în ambele direcții şi în două puncte distincte. D1 ia rute din zona RIP pe care le include în OSPF iar D2 le trimite înapoi în RIP. Atât D1 cât şi D2 vor prefera rutele ce trec prin OSPF datorită distanței administrative mai mici şi vor trimite mai departe către R1 şi R3 rutele suboptimale. Dacă R1 şi R3 preferă aceste rute în detrimentul celor interne mai bune (pe baza metricii), se ajunge la rutare suboptimală.

Pentru a remedia această problemă, există o serie de soluții. Una dintre acestea constă în a face rutele RIP reintroduse în zona OSPF „neatractive", fie prin mărirea metricii fie prin mărirea distanței administrative. O altă soluție se bazează pe filtrarea rutelor ce sunt redistribuite, pentru a nu trimite RIP înapoi în zona RIP.

7.4.3.1 Evitarea rutării suboptimale prin redistribuirea de rute neatractive

În momentul în care se face redistribuirea din OSPF în RIP, metrica rutelor introduse este setată la o valoare ridicată. Rutele către zona OSPF, fiind singurele pe care ruterele le au la dispoziție, chiar dacă au cost ridicat, vor fi folosite. Rutele suboptime, însă, vor fi înlocuite de rutele interne zonei RIP, ce au metrică mai mică.

```
D1(config)#router rip
D1(config-router)#redistribute ospf 1 metric 5

D2(config)#router rip
D2(config-router)#redistribute ospf 1 metric 5

R1#sh ip route
```

```
Gateway of last resort is 10.0.0.10 to network 0.0.0.0

     84.0.0.0/25 is subnetted, 2 subnets
C       84.34.18.0 is directly connected, Loopback0
R       84.34.18.128 [120/2] via 10.0.0.2, 00:00:15, FastEthernet0/0
     10.0.0.0/30 is subnetted, 8 subnets
C       10.0.0.8 is directly connected, FastEthernet1/0
R       10.0.0.12 [120/2] via 10.0.0.2, 00:00:15, FastEthernet0/0
C       10.0.0.0 is directly connected, FastEthernet0/0
R       10.0.0.4 [120/1] via 10.0.0.2, 00:00:15, FastEthernet0/0
R       10.0.0.24 [120/5] via 10.0.0.10, 00:00:04, FastEthernet1/0
R       10.0.0.28 [120/5] via 10.0.0.10, 00:00:04, FastEthernet1/0
R       10.0.0.16 [120/1] via 10.0.0.10, 00:00:04, FastEthernet1/0
R       10.0.0.20 [120/3] via 10.0.0.2, 00:00:15, FastEthernet0/0
R*   0.0.0.0/0 [120/5] via 10.0.0.10, 00:00:04, FastEthernet1/0
```

Se observă foarte repede rutele ce au fost primite de la OSPF (metrica = 5) şi rutele interne zonei RIP.

7.4.3.2 Evitarea rutării suboptimale prin filtrarea rutelor

Pentru a nu trimite rute înapoi în RIP şi a risca rutare suboptimală, se poate folosi un *distribute-list* pentru ca ABR-urile OSPF să nu mai promoveze către tabelele de rutare rute OSPF către reţele din zona RIP. Un exemplu de configurare pentru reţeaua de faţă:

```
D1(config)#access-list 10 deny 84.34.18.0 0.0.0.255
D1(config)#access-list 10 deny 10.0.0.0 0.0.0.3
D1(config)#access-list 10 deny 10.0.0.4 0.0.0.3
D1(config)#access-list 10 deny 10.0.0.8 0.0.0.3
D1(config)#access-list 10 deny 10.0.0.12 0.0.0.3
D1(config)#access-list 10 permit any
D1(config)#router ospf 1
D1(config-router)#distribute-list 10 in ethernet 0/0
```

Distribute-list se configurează în protocolul ce trebuie redistribuit, în cazul nostru OSPF şi filtrează toate rutele RIP ce ajung de la vecinii OSPF. Tabela de rutare a lui R1 devine:

```
R1#sh ip route
Gateway of last resort is 10.0.0.10 to network 0.0.0.0

     84.0.0.0/25 is subnetted, 2 subnets
C       84.34.18.0 is directly connected, Loopback0
R       84.34.18.128 [120/2] via 10.0.0.2, 00:00:16, FastEthernet0/0
     10.0.0.0/30 is subnetted, 8 subnets
C       10.0.0.8 is directly connected, FastEthernet1/0
R       10.0.0.12 [120/2] via 10.0.0.2, 00:00:16, FastEthernet0/0
C       10.0.0.0 is directly connected, FastEthernet0/0
R       10.0.0.4 [120/1] via 10.0.0.2, 00:00:16, FastEthernet0/0
R       10.0.0.24 [120/1] via 10.0.0.10, 00:00:15, FastEthernet1/0
R       10.0.0.28 [120/1] via 10.0.0.10, 00:00:15, FastEthernet1/0
R       10.0.0.16 [120/1] via 10.0.0.10, 00:00:15, FastEthernet1/0
R       10.0.0.20 [120/1] via 10.0.0.10, 00:00:15, FastEthernet1/0
R*   0.0.0.0/0 [120/1] via 10.0.0.10, 00:00:15, FastEthernet1/0
```

Se observă dispariţia rutelor suboptimale datorită filtrării rutelor; metrica cu care se injectează rute în RIP a fost setată înapoi la 1.

7.4.4 Policy Based Routing

Se pune acum problema implementării unor politici de rutare în reţeaua de test. Cum reţeaua dispune de două ieşiri către Internet, se doreşte folosirea uneia dintre acestea pentru traficul critic generat de către R3, cealaltă fiind folosită pentru restul traficului.

Pentru a implementa o astfel de cerință, se folosește un *route-map* prin care sunt specificate politicile dorite. În cazul de față, se va face *match* pe o serie de adrese IP sursă ce vor fi rutate apoi conform instrucțiunilor din *set*. PBR se va implementa pe ruterul E1.

```
E1(config)#access-list 10 permit 84.34.18.128 0.0.0.127
E1(config)#access-list 10 permit host 10.0.0.6
E1(config)#access-list 10 permit host 10.0.0.13
E1(config)#access-list 20 deny 84.34.18.128 0.0.0.127
E1(config)#access-list 20 deny host 10.0.0.6
E1(config)#access-list 20 deny host 10.0.0.13
E1(config)#access-list 20 permit any
E1(config)#route-map pbr 10
E1(config-route-map)#match ip address 10
E1(config-route-map)#set interface ethernet 0/2
E1(config)#route-map pbr 20
E1(config-route-map)#match ip address 20
E1(config-route-map)#set interface ethernet 0/3
```

Prima parte a *route-map-ului* va trimite către Ethernet 0/2 (ieșirea spre Internet prin E2) toate pachetele ce vin de la adresele IP ale lui R3 sau din rețeaua acestuia de *loopback*. Dacă ne-am fi oprit aici din configurare, pentru orice alt IP se va ajunge la rutare normală, existând șanse ca și alte pachete să fie trimise către ruterul E2. Pentru a nu exista astfel de probleme, se face *match* explicit pe restul traficului și acesta este trimis către ruterul E3.

Folosind **`debug ip policy`** pe ruterul E1, se poate vedea rezultatul configurărilor făcute:

```
R3#ping 23.23.23.23

E1#
*Mar  1 01:23:19.483: IP: s=10.0.0.13 (Ethernet0/1), d=23.23.23.23,
len 100, FIB policy match
*Mar  1 01:23:19.483: IP: s=10.0.0.13 (Ethernet0/1), d=23.23.23.23
(Ethernet0/2), len 100, FIB policy routed

R1#ping 23.23.23.23

E1#
*Mar  1 01:23:32.071: IP: s=10.0.0.9 (Ethernet0/0), d=23.23.23.23, len
100, FIB policy match
*Mar  1 01:23:32.071: IP: s=10.0.0.9 (Ethernet0/0), d=23.23.23.23
(Ethernet0/3), len 100, FIB policy routed
```

8 BGP

8.1 Protocolul BGP

8.1.1 Sisteme autonome

La începuturile anilor '80, ruterele care constituiau ARPANET-ul (predecesorul Internetului modern) rulau un protocol de rutare *distance-vector* cunoscut sub numele GGP (*Gateway-to-Gateway Protocol*). Pe măsură ce rețeaua creștea, arhitecții săi au constatat că acest protocol nu oferea scalabilitatea necesară.

Eric Rosen, în RFC827, identifică o serie de probleme legate de această abordare:

- *overhead*-ul dat de algoritmul de rutare crește exponențial cu creșterea dimensiunilor rețelei;
- rularea protocolului de rutare pe o varietate de platforme diferite (hardware și/sau software) face administrarea (și izolarea problemelor) aproape imposibilă;
- rezistența la schimbare crește odată cu extinderea rețelei (orice schimbare trebuie aprobată de toți administratorii).

Ca urmare, RFC827 propune o soluție inovatoare: migrarea ARPANET de la structura monolitică spre o structură modulară - un sistem de subrețele autonome interconectate. Aceste subrețele au fost denumite **sisteme autonome** (*Autonomous System* - AS).

Un AS reprezintă o colecție de rutere (și eventual alte echipamente de rețea) aflate sub un control administrativ comun. Fiecare AS este identificat printr-un număr unic la nivel global. Aceste numere (*AS Numbers*, pe scurt ASN) sunt alocate de către IANA (aceeași autoritate care alocă adresele IP).

Inițial numerele de AS erau valori întregi pe 16 biți (valori de la 0 la 65535), dar în momentul actual (2009) este în curs de desfășurare procesul de trecere la ASN pe 32 de biți. Acestea sunt reprezentate în format x.y, unde x și y sunt numere pe 16 biți. În noul format, vechile ASN (pe 16 biți) sunt notate ca 0.y.

Protocoalele de rutare discutate până acum (RIP, OSPF, EIGRP, Integrated IS-IS) rulează de obicei în interiorul unui AS, fiind cunoscute ca IGPs (*Interior Gateway Protocols*). Pentru comunicarea rutelor între sistemele autonome se folosește o nouă clasă de protocoale, așa -

numitele EGPs (*Exterior Gateway Protocols*). Din această clasă fac parte EGP (care practic nu se mai folosește astăzi) și *Border Gateway Protocol* (BGP).

La ora actuală, practic toată rutarea între ISP este realizată prin BGP versiunea 4 (denota BGPv4).

8.1.2 Când este necesar BGP?

Deseori rețelele mari sunt împărțite în mai multe domenii de rutare, fiecare rulând propriul său protocol de rutare. O greșeală frecventă este aceea de a considera că întotdeauna în astfel de cazuri este necesară rularea BGP între domeniile respective. De multe ori însă, o astfel de decizie nu face altceva decât să complice administrarea rețelei.

Din punct de vedere al interconectării lor, sistemele autonome se împart în două categorii:

- *single-homed AS* (au o singură conexiune către un alt sistem autonom). În această categorie intră majoritatea rețelelor mici (și multe dintre rețelele medii de tip enterprise);
- *multi-homed AS* (două sau mai multe conexiuni către alte sisteme autonome) - rețelele de tip ISP, sau rețele de enterprise cu conexiuni multiple la Internet.

În cazul sistemelor autonome *single-homed*, BGP nu oferă niciun fel de beneficiu. Cea mai bună soluție în acest caz este folosirea unei rute implicite (*default*), generată local sau injectată de către ISP. Complexitatea BGP devine utilă doar în cazul unui AS *multi-homed*, când prin manipularea atributelor BGP devine posibilă selectarea căilor de intrare/ieșire din AS pentru pachetele de date.

8.1.3 BGP - noțiuni de bază

Spre deosebire de IGP-urile studiate până acum, BGP nu se încadrează complet într-una din clasele *„distance-vector"* sau *„link-state"*. Cel mai corect, BGP este descris ca un *„path vector protocol"*.

Dacă în cazul IGP-urilor rutele urmăreau ruterele individuale prin care treceau pachetul, BGP adoptă o viziune de ansamblu asupra rețelei, în care fiecare AS este considerat un hop (indiferent dacă pachetele traversează un singur ruter sau 100 la trecerea prin acel AS). Pentru fiecare rețea destinație, BGP reține lista de AS-uri care vor trebui traversate de un pachet pentru a ajunge la acea rețea (*AS_PATH*), precum și *next-hop*-ul (*vectorul*, sau direcția în care trebuie trimise pachetele cu respectiva destinație). De aici numele de *„path vector protocol"*.

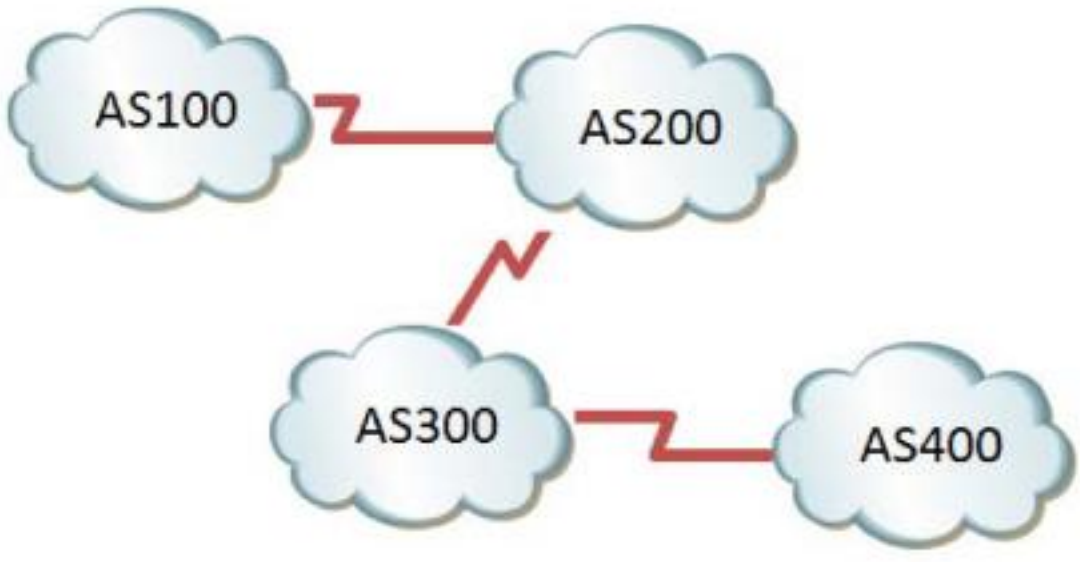

Figura 8-1: AS Path

De exemplu, dacă în Fig. 1 rețeaua 10.12.14.0/24 se află în AS100 și este anunțată în BGP, un ruter care vorbește BGP din AS400 va vedea rețeaua cu un AS_PATH având valoarea „300 200 100".

Existența *AS-Path* permite de asemenea evitarea foarte ușoară a buclelor de rutare. Dacă un ruter BGP primește un *update* care conține propriul său ASN în AS_PATH, va ignora *update*-ul respectiv.

O altă deosebire față de IGP-uri este reprezentată de modul în care BGP „descoperă" vecinii. În BGP nu există un sistem de descoperire automată a vecinilor (prin pachete *broadcast/multicast*). Vecinii sunt configurați manual pe fiecare ruter, prin specificarea adresei lor IP și a sistemului autonom din care fac parte. Comunicația cu vecinii este întotdeauna unicast, iar protocolul folosit pentru comunicație este TCP, port 179.

Sesiunea TCP este stabilită la începerea comunicației între doi vecini, moment în care se transmit tabelele de rutare complete. Ulterior, vecinii BGP vor transmite doar *update*-uri incrementale (doar lista cu modificările apărute în tabelele de rutare). Folosirea TCP garantează existența unei conexiuni sigure, și ca urmare nu este necesară trimiterea de *update*-uri complete periodice.

De asemenea, folosirea TCP și IP unicast înseamnă că vecinii BGP nu trebuie să fie întotdeauna direct conectați. O altă implicație importantă a folosirii TCP și IP unicast în conexiunile dintre vecinii BGP este faptul că realizarea unei conexiuni TCP este posibilă pentru vecinii ce nu sunt direct conectați, ci sunt situați la mai multe hopuri distanță, dar cunosc rutele necesare și pot comunica prin intermediul altor rutere (această abordare prezintă unele particularități care vor fi discutate ulterior).

Dacă în cazul IGP-urilor decizia de selecție a unei anumite căi se face pe baza unei metrici (*hop-count*, *bandwidth*, *cost*, *composite metric,* etc), în cazul BGP nu există o singură metrică. Fiecare *update* de rutare conține o serie de atribute, iar pe baza acestora (și a politicii de rutare existente în AS-ul respectiv) BGP decide care este cea mai bună cale spre diferitele destinații. O listă a celor mai frecvente atribute, precum și procesul efectiv de decizie, vor fi prezentate mai târziu în cadrul acestui capitol.

8.1.4 iBGP vs eBGP.

În funcție de sistemul autonom în care se află ruterele, conexiunile BGP se împart în două categorii:

- eBGP (external BGP) - conexiuni BGP realizate între două rutere aflate în AS-uri diferite;
- iBGP (internal BGP) - conexiuni BGP realizate între două rutere aflate în același AS.

Trebuie menționat că iBGP nu este un IGP, și nu poate fi folosit în locul unui IGP pentru rutarea în interiorul unui AS. iBGP are doar scopul de a transporta informațiile BGP între ruterele de graniță ale unui sistem autonom (numărul mare de rute cunoscute de către BGP face imposibilă transmiterea lor printr-un IGP).

Fiecare din cele două variante de BGP are propriile mecanisme de evitare a buclelor de rutare:

- eBGP - un update care conține AS-ul propriu în AS-PATH este ignorat;
- iBGP - un update primit de la un vecin iBGP nu va fi trimis mai departe către un alt vecin iBGP.

Ca o consecință a celui de-al doilea mecanism, este necesar ca într-un AS în care mai multe rutere rulează BGP conexiunile iBGP să fie *full-mesh* (fiecare ruter care vorbește BGP trebuie să stabilească o adiacență cu fiecare alt ruter BGP din același AS). Drept urmare, numărul de conexiuni iBGP va crește exponențial odată cu creșterea numărului de rutere BGP din AS. Pentru 50 de rutere BGP aflate în același AS, vor exista 50*49/2 = 1225 conexiuni iBGP. Există soluții pentru scăderea

numărului de conexiuni necesare (*route-reflectors* sau *confederations*), dar acestea necesită configurări adiționale.

8.1.5 Next-hop

O altă particularitate a BGP este reprezentată de faptul că atunci când o rută primită prin eBGP este trimisă mai departe prin iBGP, next-hop-ul rutei rămâne nemodificat. Acest lucru poate crea probleme atunci când rețelele folosite pentru conectarea la *provider* nu sunt anunțate în IGP.

De exemplu, în figura 2, rețeaua clientului este conectată la ISP prin rețeaua 172.27.11.0/30. Orice rută primită de client de la ISP va fi primită de R3 cu next-hop 172.27.11.1. În momentul în care ruta este trimisă de R3 către R1 prin iBGP, R3 nu se anunță pe sine ca next-hop, ci păstrează next-hop-ul original. Ca urmare, R1 va primi rutele cu next-hop 172.27.11.1.

Ținând cont că IGP-ul folosit în rețeaua clientului (OSPF, de exemplu) nu va rula și pe legătura dintre client și ISP, rețeaua 172.27.11.0/30 nu va apărea în tabelele de rutare de pe ruterele R1 și R2. Din acest motiv, R1 nu va ști cui trebuie să trimită pachetele destinate rețelelor externe (știe că trebuie să le trimită către 172.27.11.1, dar nu știe cum să ajungă la 172.27.11.1). Pentru a evita această problemă, ruterele nu instalează o rută primită prin BGP în tabela de rutare decât dacă next-hop-ul rutei este valid și au o rută către el.

Pentru rezolvarea acestei probleme există două variante:

- anunțarea rețelelor de interconectare în IGP;
- configurarea ruterelor de *edge* să schimbe next-hop-ul la anunțarea rutelor în iBGP (folosind comanda **`neighbor`** cu parametrul **`next-hop-self`**, descrisă în capitolul de configurare).

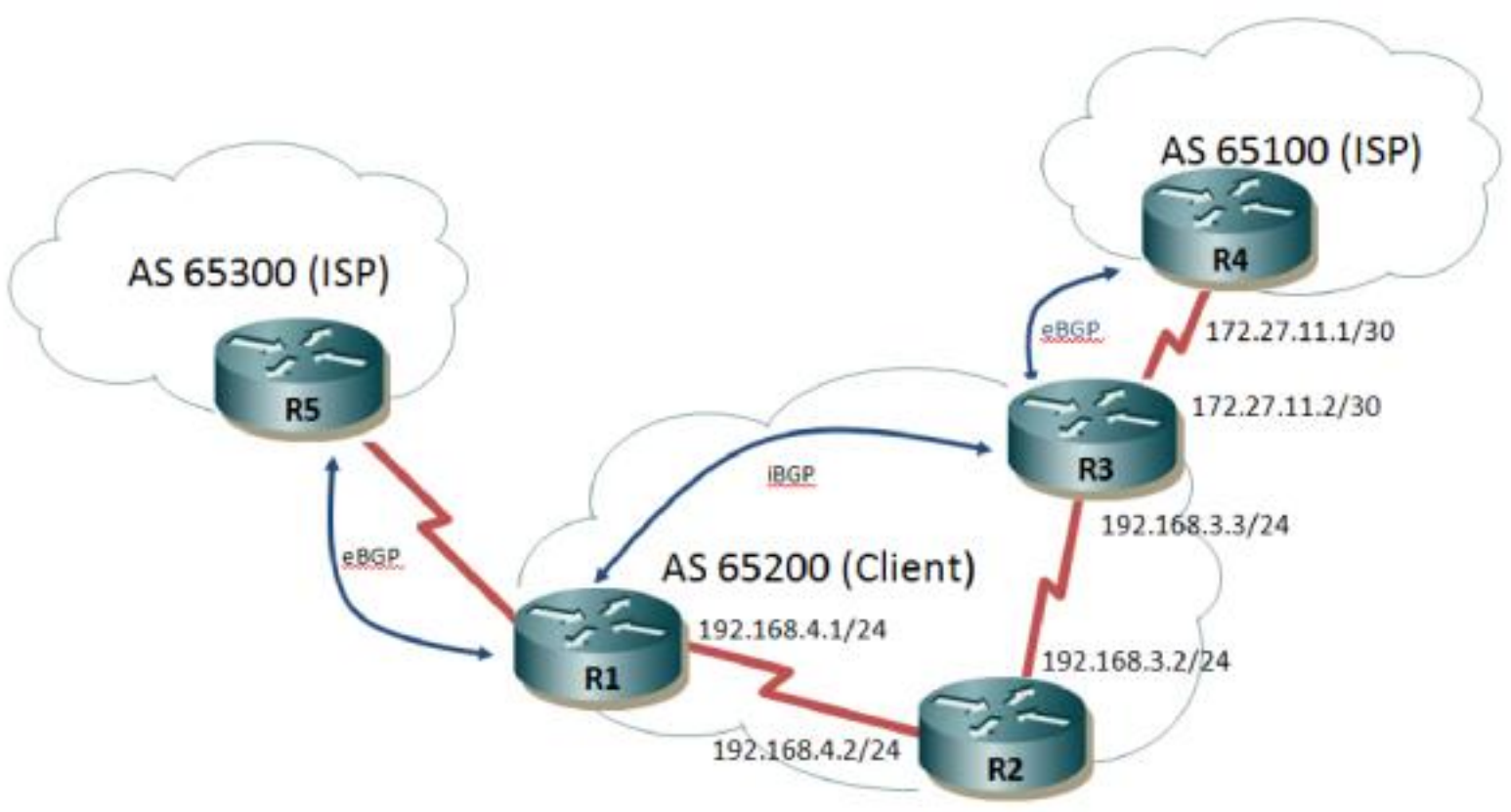

Figura 8-2: eBGP și iBGP

Figura 2 prezintă de asemenea o altă problemă, specifică AS-urilor de tranzit. Un AS de tranzit (*transit AS)* este conectat la 2 sau mai multe alte AS-uri, și transportă traficul dintre acestea. De obicei, AS-urile furnizorilor de Internet (*ISPs*) sunt AS-uri de tranzit.

Într-un AS de tranzit, iBGP are scopul de a comunica informații între ruterele de graniță, pentru ca aceste informații să fie transmise mai departe către AS-urile vecine. Dar faptul că vecinii de iBGP nu sunt întotdeauna direct conectați poate crea o situație particulară.

De exemplu, să presupunem că în AS 65100 se află rețeaua 10.0.0.0/8. Prefixul respectiv este anunțat în BGP și trimis mai departe către R3, R1 și R4. Toate acestea vor instala prefixul în tabela de rutare (inclusiv R1, care îl va instala cu adresa R3 ca next-hop).

În momentul în care din AS 65300 se trimite un pachet către un IP din 10.0.0.0/8, pachetul va ajunge la R1. R1 verifică tabela de rutare, și găsește adresa R3 (192.168.3.3) ca next-hop. Dar adresa respectivă nu este direct conectată, motiv pentru care R1 va face o nouă parcurgere a tabelei de rutare - de data aceasta căutând next-hop-ul către 192.168.3.3. Este găsită adresa R2 (192.168.4.2), care este direct conectată. R1 încapsulează pachetul, și îl trimite către R2.

Pachetul ajunge la R2, iar acesta va căuta în tabela sa de rutare o rută către destinație. Dar R2 nu rulează decât IGP-ul din AS 65200 (OSPF, în exemplul nostru), nu și BGP. Ca urmare, R2 nu are nici o rută spre destinație, și va face *discard* la pachet. Astfel, deși toate tabelele de rutare BGP par corecte, R2 acționează efectiv ca o „gaură neagră" (*black hole*) în care dispare traficul de tranzit.

Teoretic, o soluție pentru această problemă ar fi reprezentată de redistribuția rutelor din BGP în IGP. După cum am menționat însă mai devreme, aceasta nu este o soluție scalabilă (numărul foarte mare de rute transportate de BGP nu poate fi suportat de nici unul din IGP-urile actuale). Ca urmare, este necesară rularea iBGP pe toate ruterele din AS, și ca urmare realizarea unui *full-mesh* iBGP între toate ruterele din AS, cu toate problemele discutate mai devreme. De menționat este că în implementările reale a AS-urilor de tranzit se folosește MPLS pentru a evita rularea BGP pe toate ruterele *ISP*-ului.

8.1.6 Tabele BGP

În deciziile de rutare pe care protocolul BGP le ia sunt folosite trei tabele importante: de adiacență, de rutare și tabela BGP (tabela de topologie).

Pentru ca BGP să realizeze o adiacență trebuie configurat explicit pentru fiecare vecin cu care se dorește acest lucru. Adiacențele BGP se creează folosind portul 179 TCP și se mențin folosind mesaje TCP *keepalive* o dată la 60 de secunde. După stabilirea adiacenței, ruterele BGP fac schimb de informații de rutare din tabela de rutare. Toate rutele primite de BGP sunt păstrate în tabela BGP însă doar cea mai bună rută este promovată în tabela de rutare.

Ruta cea mai bună este aleasă după un proces de decizie de 11 pași, descris în cele ce urmează. Rutele eBGP au o distanță administrativă de 20 în timp ce rutele iBGP au 200.

8.2 Implementarea politicilor BGP

După cum s-a menționat și în capitolul anterior, BGP este un protocol folosit strict pentru implementarea unor politici de rutare pentru o organizație. Față de orice alt protocol de rutare ce este interesat de găsirea celui mai scurt drum către destinație, BGP poate controla direcționarea traficului către anumite destinații și chiar către AS-ul propriu.

Pentru a putea configura politici BGP se folosesc *route-map*-uri care, în funție de anumite rețele destinație, modifică atribute BGP ce pot influența rute.

8.2.1 Atribute BGP

Spre deosebire de IGP-uri, BGP nu folosește o singură m etrică pentru a decide cea mai bună cale spre o anumită destinație. În schimb, BGP se bazează pe așa numitele *atribute*. Fiecare *update* BGP conține, pe lângă prefixul destinație, și o serie de atribute care caracterizează acel prefix.

Atributele sunt împărțite în următoarele categorii:

- Well-known mandatory

- Well-known discretionary
- Optional transitive
- Optional non-transitive

Un atribut *well-known* trebuie să fie recunoscut de orice implementare BGP care respectă standardele, în timp ce un atribut *optional* poate să nu fie recunoscut de unele implementări (un atribut *optional* care nu este recunoscut va fi pur și simplu ignorat).

Atributele well-known pot fi *mandatory* (trebuie să fie incluse întotdeauna, în orice *update* BGP), sau *discretionary* (pot lipsi din unele *update*-uri).

Dacă un atribut opțional este *transitive*, un ruter BGP care nu îl recunoaște va trimite atributul mai departe, nemodificat, către vecinii săi BGP. Un atribut *optional non-transitive*, în schimb, nu va fi trimis mai departe de către un ruter care nu îl recunoaște.

Atributele opționale care sunt recunoscute de către un ruter BGP pot fi trimise sau nu mai departe către vecinii săi, în funcție de semnificația atributului.

Atributele BGP specificate în RFC1771 sunt următoarele:

- ORIGIN - Well-known mandatory
- AS_PATH - Well-known mandatory
- NEXT_HOP - Well-known mandatory
- LOCAL_PREF - Well-known discretionary
- ATOMIC_AGGREGATE - Well-known discretionary
- AGGREGATOR - Optional transitive
- COMMUNITY - Optional transitive
- MULTI_EXIT_DISC(MED) - Optional nontransitive
- ORIGINATOR_ID - Optional nontransitive
- CLUSTER_LIST - Optional nontransitive

8.2.1.1 ORIGIN

Atributul ORIGIN este un atribut *well-known mandatory* care specifică originea update-ului de rutare. Variantele posibile pentru acest atribut sunt:

- IGP (notat „i") - prefixul a fost învățat printr-un IGP (RIP, OSPF, EIGRP, etc), și injectat ulterior în BGP prin comanda **`network`**;
- EGP (notat „e") - prefixul a fost învățat prin EGP. Cum EGP nu mai este folosit astăzi, nu se mai întâlnesc prefixe cu ORIGIN „e";
- Unknown (notat „?") - sunt prefixe învățate din alte surse. Rutele care sunt redistribuite în BGP (prin comanda *redistribute*) sunt marcate cu ORIGIN „?".

Dacă un ruter BGP primește mai multe rute către aceeași destinație, cu ORIGIN diferit, va prefera rutele în ordinea de mai sus (sunt preferate rutele „i", apoi „e", ultimele sunt „?"). Notație: i > e > ?

8.2.1.2 AS_PATH

Atributul AS_PATH este un atribut *well-known mandatory* care conține lista de ASN-uri care vor trebui traversate pentru a ajunge la destinația specifică în update. Fiecare ruter BGP care trimite ruta către un vecin eBGP va adăuga în listă propriul său ASN. Adăugarea se face la începutul listei (*prepending*), astfel încât întotdeauna lista din AS_PATH va începe cu cel mai apropiat AS, și se va încheia cu AS-ul în care a originat update-ul.

Reluând exemplul din Fig.1, în care o rută din AS100 este anunțată prin BGP:

- în momentul în care ruta este anunțată în BGP din AS100, ruterul care o anunță își adaugă propriul ASN în AS_PATH;

- Pentru fiecare ruter care anunță ruta către un vecin eBGP, ASN-ul ruterului va fi adăugat la începutul listei. Astfel, un ruter BGP din AS 300 va vedea ruta cu AS_PATH=200 100, iar în AS 400 AS_PATH va fi 300 200 100.

Ca o metodă de evitare a buclelor în rutarea inter-AS, un ruter nu va accepta un update eBGP care conține propriul ASN în AS_PATH. Acesta este unul din atributele importante care se pot manipula din configurații.

8.2.1.3 NEXT_HOP

Atributul NEXT_HOP (*well-known, mandatory*) conține adresa IP a următorului ruter pe calea către destinație. Trebuie menționat că adresa IP din NEXT_HOP nu aparține întotdeauna unui ruter vecin:

- Dacă ruterul care anunță prefixul și ruterul care îl primește sunt în AS-uri diferite (eBGP), NEXT_HOP va fi setat la adresa IP a ruterului care anunță prefixul;
- Dacă ruterul care anunță prefixul și ruterul care îl primește sunt în același AS (iBGP), și prefixul aparține acelui AS, NEXT_HOP este adresa ruterului care a anunțat ruta;
- Dacă ruterul care anunță prefixul și ruterul care îl primește sunt în același AS (iBGP), și prefixul aparține unui alt AS, NEXT_HOP va fi adresa ruterului extern care a anunțat ruta. Un alt mod de a formula această regulă este următorul: „La trecerea din eBGP în iBGP, NEXT_HOP nu se modifică".

8.2.1.4 LOCAL_PREF

Atributul LOCAL_PREF (*Local Preference*) este un atribut *well-known discretionary* folosit doar în update-uri iBGP (nu se trimite către alte AS-uri). Acest atribut este folosit pentru a comunica preferința pentru o anumită rută. Dacă un ruter BGP primește mai multe rute către aceeași destinație cu LOCAL_PREF diferite, va prefera ruta cu LOCAL_PREF mai mare. Acesta este unul din atributele importante care se pot manipula din configurații.

8.2.1.5 MULTI_EXIT_DISC (MED)

LOCAL_PREF nu afectează decât traficul care părăsește un AS. MULTI_EXIT_DISC (pe scurt MED) este un atribut *optional non-transitive* care permite unui AS să influențeze traficul său de intrare. MED conține o valoare pe 16 biți, cunoscută și ca „INTER_AS Metric" (în terminologia BGPv2 și BGPv3). Un ruter care primește mai multe rute care diferă prin MED va prefera ruta cu MED mai mic (spre deosebire de LOCAL_PREF, unde o valoare mai mare este preferată). Aceasta deoarece MED este considerat o metrică, și ca la orice metrică, valorile mai mici sunt considerate mai bune.

Folosirea MED pentru manipularea traficului de intrare în AS are o serie de limitări, principala fiind dată de faptul că MED nu se transmite mai departe de AS-ul vecin. Ca urmare, MED nu poate fi utilizat în cazul unui AS legat la 2 sau mai multe AS-uri diferite (de exemplu o rețea de enterprise legată la 2 ISP), deoarece nici unul dintre AS-urile vecine nu va ști care este valoare MED primită de celelalte AS-uri. Din acest motiv, în general se preferă alte metode pentru influențarea traficului de intrare, bazate pe manipularea AS_PATH.

8.2.1.6 COMMUNITY

Atributul COMMUNITY este un atribut *optional transitive* utilizat pentru a simplifica aplicarea de politici de rutare. Inițial un atribut specific Cisco, a fost standardizat în RFC1997. Atributul identifică un prefix ca făcând parte dintr-o anumită „comunitate" - un set de prefixe caracterizate prin proprietăți comune. De exemplu, un ISP poate seta un anumit COMMUNITY pentru toate rutele

clienților săi, urmând ca ulterior să aplice valori particulare pentru LOCAL_PREF și MED pe baza acestui atribut.

Atributul are ca valoare un set de 4 octeți. Conform RFC 1997, acesta este de forma AA:NN - primii 2 octeți identifică AS-ul, iar ultimii 2 sunt identificatorul comunității.

Există câteva valori predefinite pentru COMMUNITY, cu semnificație particulară:

- NO_EXPORT - rutele marcate cu această comunitate nu sunt anunțate către vecini eBGP (nu părăsesc AS-ul), cu excepția vecinilor care fac parte din aceeași confederație;
- NO_ADVERTISE - rutele marcate astfel nu sunt anunțate către nici un fel de vecini BGP (fie ei iBGP sau eBGP);
- LOCAL_AS - rutele marcate astfel nu sunt anunțate către nici un fel de vecini eBGP (nici către vecini din aceeași confederație).

8.2.1.7 ORIGINATOR_ID, CLUSTER_LIST

ORIGINATOR_ID și CLUSTER_LIST sunt atribute *optional non-transitive* folosite de către route-reflectors pentru prevenirea buclelor de rutare. Route-reflector-ii reprezintă rutere BGP cu o configurație specială, folosite pentru a scădea numărul de adiacențe iBGP necesare într-un sistem autonom.

8.2.1.8 WEIGHT

Deși nu este un atribut BGP propriu-zis, WEIGHT (sau *„Administrative weight"*) reprezintă un alt factor care influențează decizia BGP. Parametrul WEIGHT este specific Cisco, și se aplică doar pe ruterul pe care este configurat (nu este comunicat către alte rutere). Valoarea sa este un număr pe 16 biți (între 0 și 65.535), valorile mai mari fiind preferate. Implicit, rutele generate local au WEIGHT 32.768, iar cele primite de la un vecin au WEIGHT 0. Acesta este unul din atributele importante care se pot manipula din configurații.

8.2.2 Procesul de decizie BGP

După cum s-a menționat mai devreme, fiecare prefix primit prin BGP va fi însoțit de o serie de atribute. În momentul în care un ruter BGP primește mai multe *update*-uri referitoare la aceeași destinație, va trebui să aleagă cea mai bună cale spre destinația respectivă. Pentru aceasta, ruterul compară atributele în ordinea următoare:

1. WEIGHT. Valorile mai mari sunt preferate;
2. LOCAL_PREF. Valorile mai mari sunt preferate;
3. Locally originated - se preferă rutele originate local;
4. AS_PATH - se preferă un AS_PATH mai scurt;
5. ORIGIN. Ordinea preferinței este: i>e>?;
6. MED. Se preferă valoarea cea mai mică. Această comparație se va face doar dacă rutele comparate provin din același AS;
7. Se preferă rutele primite prin eBGP față de cele primite prin iBGP;
8. IGP to NEXT_HOP. Se preferă ruta cu metrica IGP cea mai mică până la NEXT_HOP;
9. Dacă toți parametrii de mai sus sunt egali și BGP este configurat să instaleze mai multe rute în tabela de rutare (prin comanda **`bgp maximum-paths`**), toate rutele se vor instala în tabelă. Dacă BGP nu este configurat pentru multipathing, se merge mai departe;
10. Oldest external route. Dacă ambele rute sunt externe, se preferă ruta cea mai veche. Acest pas are scopul de a minimiza *route-flapping*-ul (modificările repetate ale rutei către aceeași destinație);
11. Ruter ID. Se alege ruta venită de la ruterul cu BGP RouterID cel mai mic.

Pașii sunt parcurși în ordinea menționată, până când se alege o singură rută ca fiind cea mai bună. Un atribut nu va fi comparat decât dacă toate atributele listate înaintea lui au valori egale (de exemplu, două rute nu vor fi comparate pe baza AS_PATH decât daca au WEIGHT și LOCAL_PREF egale, și nici una nu a fost originată local).

Trebuie menționat că odată parcurs procesul de decizie, un ruter BGP va anunța către vecini *doar cea mai bună cale* (nu toate căile posibile).

8.2.3 Scalarea BGP

Problema din cauza căreia se creează găuri negre de rutare se datorează faptului că nu există un *full-mesh* iBGP în AS. Motivul pentru care există această problemă este regula de prevenire a buclelor în iBGP. Conform acestei reguli, un ruter ce rulează iBGP nu are voie să transmită unui vecin iBGP o rută primită prin iBGP.

Această regulă poate fi depășită printr-o metodă de scalare ce presupune folosirea unor rutere speciale în implementarea BGP numite *route-reflectors*. Aceste rutere au posibilitatea de a defini anumiți clienți BGP pentru care să realizeze o excepție de la regula de prevenire a buclelor iBGP.

Ruterele dintr-un AS se vor împărți astfel în 3 categorii

- Ruterele *route-reflectors* – ruterele care „reflectă" rute iBGP la clienții desemnați;
- Ruterele clienți *route-reflectors* – ruterele care primesc rute prin iBGP de la *route-reflectors*;
- Ruterele normale iBGP – rutere care trebuie să fie conectate în *full-mesh* pentru a nu genera găuri negre de rutare.

Folosirea acestei metode de scalare presupune ca grupul de clienți *route-reflectors* să aibă doar o relație de adiacență cu ruterul *route-reflector*, deoarece acesta le va trimite toate rutele pe care le obține fie prin eBGP sau iBGP. În același timp, ruterele ce nu sunt clienți route-reflectors vor trebui să aibă relații iBGP *full-mesh* atât între ei cât și cu ruterul *route-reflector* pentru a preveni posibilele găuri negre.

Această metodă de scalare BGP a avut parte de o implementare extrem de elegantă în protocol. Deoarece în toată strategia de mai sus, doar *route-reflector*-ul este cel ce încalcă regula iBGP pentru bucle, toată configurația se realizează pe acesta. Configurația constă în definirea unei liste de clienți pentru care *route-reflector*-ul să facă excepția mai sus menționată.

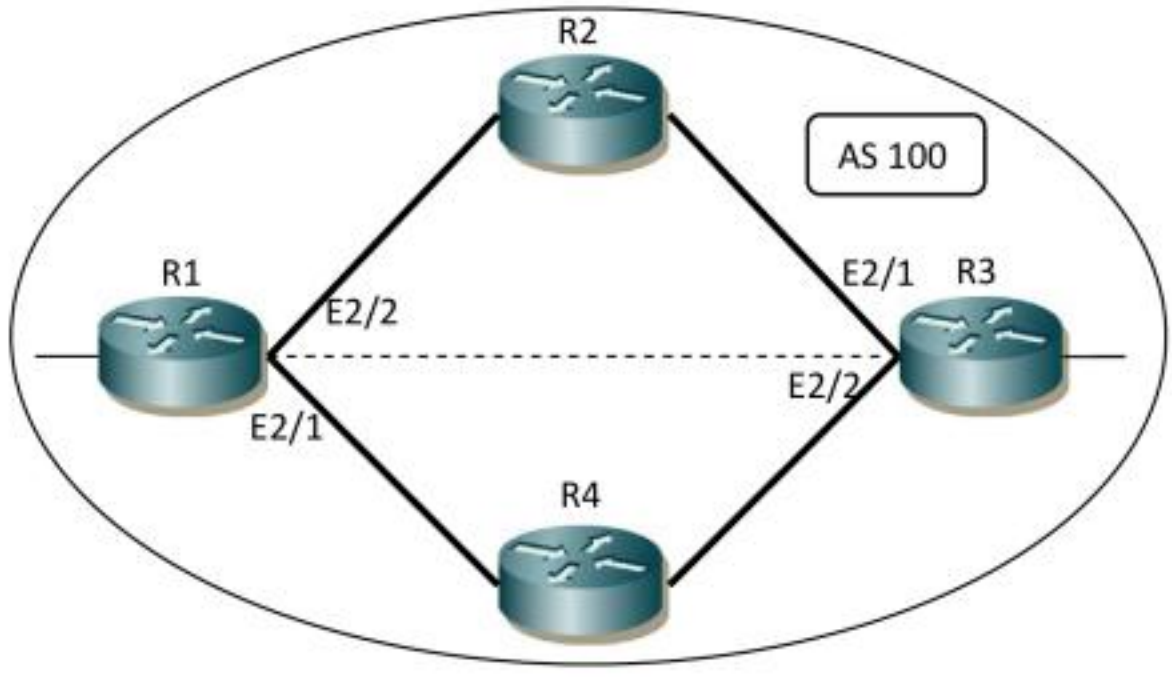

Figura 8-3: Route Reflectors

În topologia de mai sus se consideră următoarele adiacențe iBGP: R1 – R2, R1 – R4, R2 –R3, R3-R4. Ruterele R1și R3 sunt ruterele care comunică cu alte AS -uri în afara AS 100. Aparent există

adiacențe care să unească toate ruterele din AS însă pentru că nu există *full-mesh* unele rutere nu vor afla toate informațiile din AS.

Când un *update* eBGP va veni la ruterul R1 acesta îl va trimite la ruterele cu care are sesiuni iBGP. Astfel R2 și R4 vor primi *update*-ul prin iBGP însă nu vor putea să îl dea mai departe lui R3 deoarece ar încălca regula de evitare a buclelor.

Configurând pe R2 ca *route-reflector* și pe R3 ca și client *route-reflector*, R2 va putea trimite toate rutele primite de la un non-client (R1) la toți clienții săi (R3).

În încheierea acestui subcapitol se va atrage atenția asupra unei greșeli des întâlnite în configurațiile de *route-reflectors*. Când se folosește această metodă toate ruterele care sunt non-clienți trebuie să fie conectați în *full-mesh*. Deoarece un *route-reflector* trimite toate rutele de la un non-client doar la clienții săi, neatenția în configurare poate să duca la rutere non-clienți ce nu cunosc întreagă informație de rutare BGP.

8.3 Configurarea protocolului

Deși BGP este unul dintre cele mai complexe protocoale din lumea rețelelor de calculatoare, el este relativ ușor de configurat din punct de vedere al sintaxei. Motivul principal este faptul că Cisco a grupat în IOS cea mai mare parte dintre toate configurările BGP într-o singură comandă: **neighbor**.

Pentru a porni procesul BGP pe un ruter trebuie dată aceeași comandă ca și în cazul oricărui alt protocol de rutare: **router**. În cazul BGP această comandă primește ca și parametru atât cuvântul cheie **bgp** cât și un număr de AS. Cel din urmă argument are rolul de a îi indica procesului de BGP în ce AS se află. Ca exemplu, în *output*-ul de mai jos, procesul BGP este pornit pentru AS-ul 100:

```
R1(config)#router bgp 100
R1(config-router)#
```

Realizarea unei configurații BGP poate fi în general împărțită în două categorii: realizarea adiacenței între vecini și politici BGP. Configurațiile avansate BGP introduc multe alte cerințe de scalabilitate sau configurare cu alte protocoale (MPLS, IPv6 etc.), însă această introducere va trata doar configurarea adiacențelor și a politicilor de bază.

8.3.1 Configurarea adiacențelor BGP

Adiacențele BGP pot fi de 2 tipuri: iBGP sau eBGP. Se va începe cu iBGP. Presupunând o rețea precum cea din figura de mai jos, cerința este de a realiza o sesiune iBGP între ruterul R1 și R2. De asemenea se va presupune că între cele 4 rutere există deja conectivitate prin IGP-ul OSPF.

Cea mai rapidă metodă de a realiza sesiunea iBGP între cele 2 rutere dorite este alegerea unuia dintre IP-urile de pe o interfață și folosirea acestuia în comanda **neighbor**.

```
R1(config-router)#neighbor 11.0.0.1 remote-as 100
R3(config-router)#neighbor 10.0.0.1 remote-as 100
```

După cum se observă, ambele comenzi au folosit pe lângă IP-ul vecinului BGP și cuvântul cheie **remote-as** urmat de un număr de AS. Acest număr de AS împreună cu numărul de AS dat ca parametrul comenzii **router** îi spune protocolului dacă sesiunea ce se dorește stabilită este iBGP sau eBGP. Dacă numerele de AS sunt identice în cele 2 comenzi, protocolul BGP va înțelege că ruterele vecine se află în același AS și va porni o sesiune iBGP. În caz contrar o adiacență eBGP va fi creată.

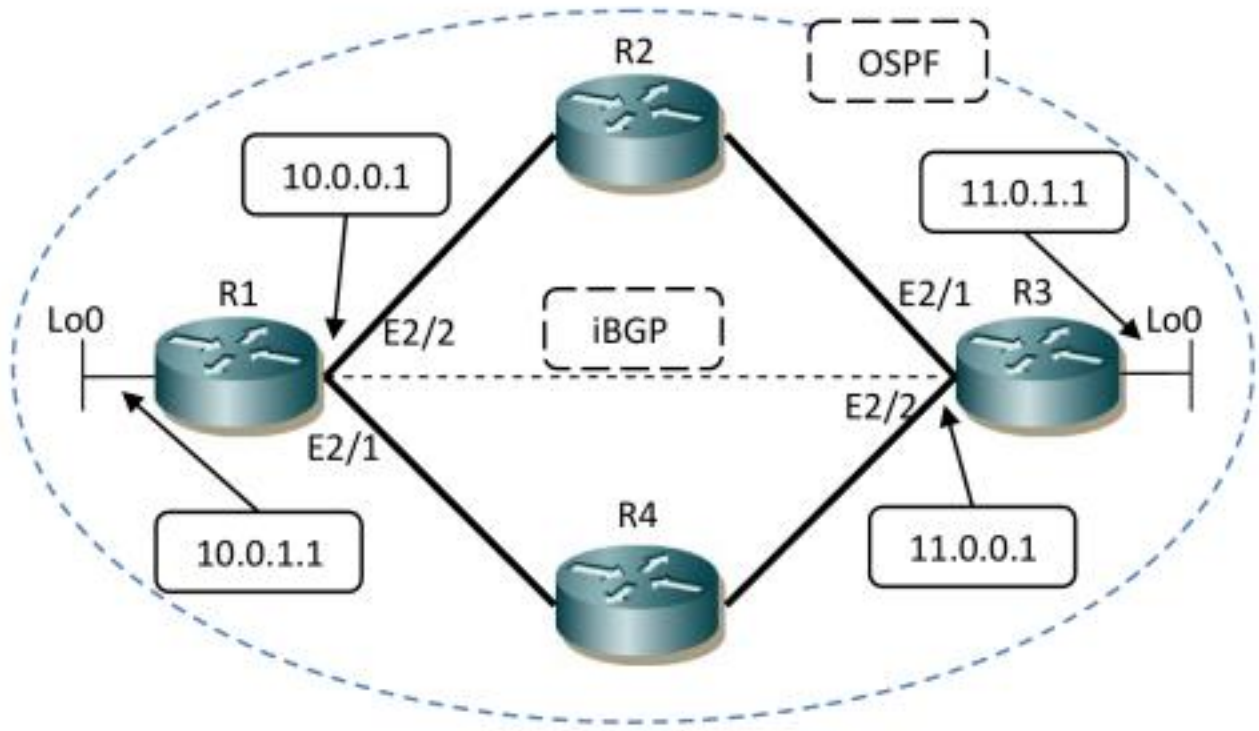

Figura 8-4: Adiacențe iBGP

Notă: pentru a se stabili o sesiune de adiacență BGP trebuie să existe conectivitate între ruterul pe care se dă comanda **`neighbor`** *și adresa IP a vecinului. În cazul de mai sus această adiacență este asigurată de protocolul OSPF.*

Odată cu executarea comenzilor de mai sus sesiunea iBGP va fi creată cu succes. O posibilă problemă poate apărea însă, dacă una dintre interfețele celor două rutere va începe să facă *flapping*[1] sau se va deconecta. Acest lucru se poate întâmpla din multiple motive, de la condițiile de mediu sau o defecțiune hardware, până deconectarea accidentală a unor cabluri. Oricare din aceste acțiuni va avea ca rezultat distrugerea adiacenței BGP. La oricare alt protocol de rutare acest lucru ar fi ușor remediat deoarece adiacența s-ar reconstrui repede iar cantitatea de rute trimisă nu ar fi atât de mare. BGP poate ajunge la o tabelă de rutare de peste 300 000 de rute, în care restabilirea unei adiacențe are costuri mari traduse în *downtime*.

Pentru a preveni astfel de situații, o recomandare este folosirea interfeței de *loopback* de pe fiecare ruter pentru a stabili adiacențele. Motivul este faptul că interfețele de loopback există doar în software și nu sunt supuse problemelor mediului fizic. Atenție însă, acest lucru va funcționa doar dacă adresa IP a interfeței de *loopback* a vecinului BGP este cunoscută în tabela de rutare printr-un protocol IGP sau printr-o rută statică. În cazul de mai sus, daca administratorul dorește realizarea unei sesiuni BGP folosind interfețele de *loopback*, va trebui să introducă rețelele de *loopback* în OSPF.

```
R1(config)#router ospf 1
R1(config-router)#redistribute connected route-map all_loopbacks subnets
R1(config-router)#exit
R1(config)#route-map all_loopbacks permit 10
R1(config-route-map)#match interface Loopback0
R1(config-route-map)#exit
R1(config)#router bgp 100
R1(config-router)# neighbor 11.0.1.1 remote-as 100

R3(config)#router ospf 1
R3(config-router)#redistribute connected route-map all_loopbacks subnets
R3(config-router)#exit
R3(config)#route-map all_loopbacks permit 10
R3(config-route-map)#match interface Loopback0
R3(config-route-map)#exit
```

[1] Fenomenul în care o interfață oscilează între stările *up/down*.

```
R3(config)#router bgp 100
R3(config-router)# neighbor 10.0.1.1 remote-as 100
```

Siguranța adiacenței ce este dorită prin crearea sesiunii iBGP peste loopback-uri introduce însă unele complicații în configurare. După comenzile date mai sus se va observa că adiacența iBGP nu s-a realizat. Dacă s-ar da comanda **debug ip bgp** s-ar observa că IOS raportează o nepotrivire între adresa sursă a pachetelor BGP și adresa de *neighbor* cu care vecinul respectiv este cunoscut.

Această problema este normală și apare din cauza faptului că fiecare vecin în mod implicit folosește adresa IP de pe interfața de ieșire a pachetelor ca și adresă IP sursă în pachetele BGP. Când pachetele ajung la ruterul vecin acesta le refuză deoarece se așteptă să vadă IP -ul configurat în comanda **neighbor** (IP-ul interfeței de loopback a vecinului) în câmpul de IP sursă al pachetului.

Pentru a rezolva această problemă și a obține adiacența iBGP trebuie dată comanda **neighbor** cu argumentul **update-source** și interfața a cărei adresă IP să fie folosită ca și adresă sursă în pachete (în cazul de față, Loopback0).

```
R1(config-router)#neighbor 11.0.1.1 update-source Lo0
R3(config-router)#neighbor 10.0.1.1 update-source Lo0
```

iBGP mai are doar un singur mic detaliu de configurare și este gata pentru funcționare în majoritatea topologiilor BGP. Deși sesiunea iBGP funcționează corect, la introducerea unor rute din eBGP în iBGP va apărea o problemă de conectivitate.

Se va presupune că ruterul R1 din figura de mai sus rulează și eBGP cu un ruter numit R_ext din alt AS și că primește de la acesta rute BGP. Când eBGP transmite rute, schimbă automat informația de *next-hop* pentru rutele respective. Acest lucru înseamnă că atunci când R_ext dorește să transmită rute către R1, va schimba adresa *next-hop*-ului la adresa de pe interfața sa de ieșire către R1. Prin această acțiune R_ext îi indică lui R1 faptul că rețelele din rutele tran smise se găsesc undeva dincolo de el. Comportamentul este cât se poate de normal pentru orice protocol de rutare.

iBGP nu schimbă însă această informație la transmiterea rutelor. Astfel, când R1 va trimite lui R3 rutele învățate de la R_ext, *next-hop*-ul va rămâne neschimbat la adresa IP a lui R_ext. Acest lucru înseamnă că R3 nu va putea să instaleze niciodată aceste rute în tabela de rutare deoarece nu are conectivitate către *next-hop*. Prima reacție în scopul rezolvării acestei probleme ar putea fi configurarea unei rute statice pe R3 pentru rețeaua dintre R1 și R_ext. Deși această rută statică ar rezolva problema, nu este o soluție scalabilă și nici o practică indicată. Un ruter din interiorul unui AS nu ar trebui să aibă rute statice către rețele din exteriorul AS-ului; pentru acest lucru exista BGP.

Soluția cea mai bună este configurare iBGP pentru a avea același comportament de schimbare a next-hop-ului ca și eBGP. Acest lucru se realizează prin comanda **neighbor** cu parametrul **next-hop-self**.

```
R1(config-router)#neighbor 11.0.1.1 next-hop-self
R3(config-router)#neighbor 10.0.1.1 next-hop-self
```

8.3.2 Configurarea adiacențelor eBGP

În cazul eBGP configurațiile sunt asemănătoare cu iBGP însă nu există complicații introduse de informația de next-hop deoarece eBGP schimbă în mod implicit adresa *next-hop* la transmiterea unei rute.

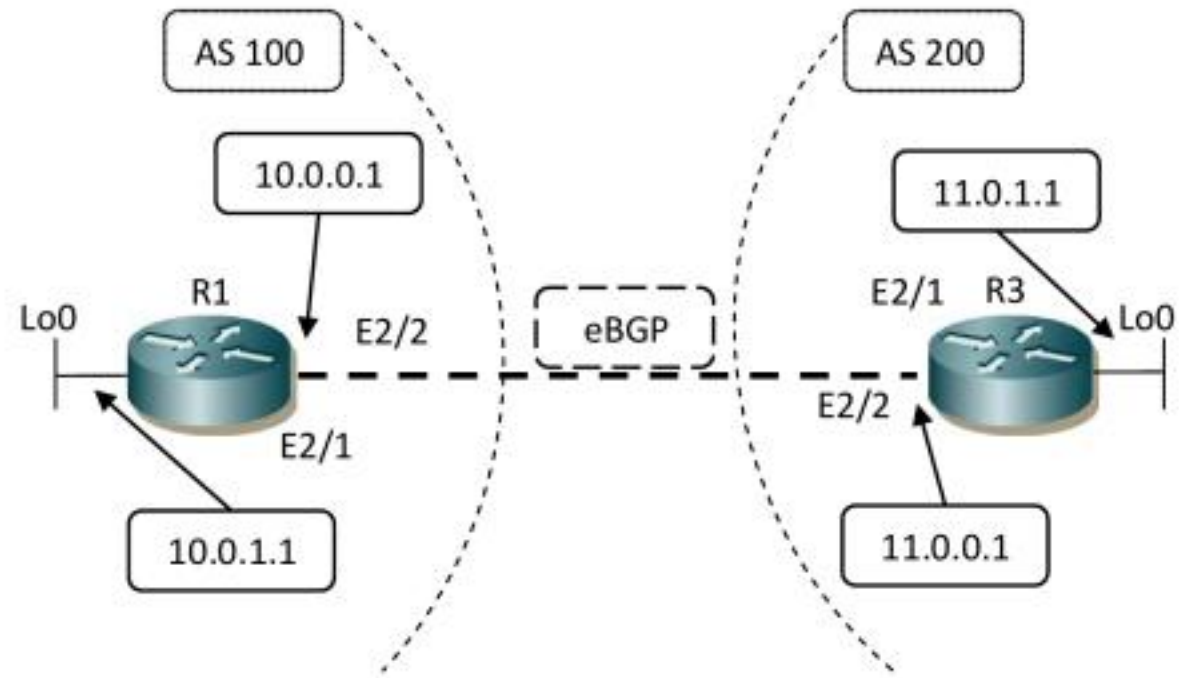

Figura 8-5: Configurarea adiacenței eBGP

Se vor folosi aceleași rutere ca protagoniști ai scenariului dar de data aceasta în AS-uri diferite.

```
R1(config)#router bgp 100
R1(config-router)#neighbor 11.0.0.1 remote-as 200
R3(config)#router bgp 200
R3(config-router)#neighbor 10.0.0.1 remote-as 100
```

Pentru ca numărul de AS dat în comanda **`router`** este diferit de cel folosit în comanda **`neighbor`**, IOS va porni o sesiune de eBGP. Sesiunea va funcționa în această formă fără orice altă modificare.

Există însă cazuri în care cele 2 rutere de mai sus sunt legate prin 2 cabluri pentru redundanță. Întrebarea ce se pune în acest caz este: ce IP va fi folosit în fiecare capăt pentru crearea adiacenței? Dacă se va folosi oricare dintre IP-urile de pe legăturile fizice se va risca pierderea adiacenței ca și în cazul iBGP din aceleași motive enumerate în subcapitolul anterior. Soluția va fi di n nou folosirea unei interfețe de loopback. Deoarece între AS-uri nu rulează nici un protocol IGP, fiecare ruter va trebui să definească o rută statică pentru rețeaua în care se găsește *loopback*-ul vecinului.

```
R1(config)#ip route 11.0.1.0 255.255.255.0 11.0.0.1
R1(config)#router bgp 100
R1(config-router)#neighbor 11.0.1.1 remote-as 200
R1(config-router)#neighbor 11.0.1.1 update-source Loopback 0
```

```
R3(config)#ip route 10.0.1.0 255.255.255.0 10.0.0.1
R3(config)#router bgp 200
R3(config-router)#neighbor 10.0.1.1 remote-as 100
R3(config-router)#neighbor 10.0.1.1 update-source Loopback 0
```

În acest moment, atât rutele statice cât și vecinii au fost configurați corect. eBGP nu va ridica însă adiacența cu vecinul său. Problema constă în modul în care eBGP a fost implementat în cod. În mod implicit, programatorii BGP au presupus că o adiacență eBGP va fi întotdeauna făcută între rutere direct conectate căci între AS-uri nu rulează niciun protocol IGP. Astfel toate pachetele de stabilire de adiacență eBGP au un TTL implic it cu valoarea „1”. Dacă se folosesc interfețele de Loopback pentru stabilirea adiacențelor, TTL -ul va fi decrementat la rutarea spre loopback iar pachetul va fi aruncat.

Pentru a rezolva această ultimă problemă de adiacență se folosește comanda **neighbor** cu parametrul **ebgp-multihop** urmat de valoarea TTL pe care pachetele de adiacență ar trebui să o aibă.

```
R1(config-router)#neighbor 11.0.1.1 ebgp-multihop 2
R3(config-router)#neighbor 10.0.1.1 ebgp-multihop 2
```

8.3.3 Configurarea politicilor BGP

Manipulând atributele, o organizație poate controla fluxul de date care intră și care iese din propriul AS. Există multe moduri prin care aceste politici se pot realiza, cele mai populare fiind: modificarea *local preference*, *AS-PATH prepend*, modificarea *MED*, modificarea *weight*. În capitolul de studiu de caz se vor prezenta aceste politici aplicate pe o topologie întâlnită în implementări reale.

Indiferent de tipul de atribut ce se dorește modificat, BGP are nevoie de folosirea unui *route-map* pentru realizarea acestui lucru.

```
R1(config-router)#neighbor 11.0.1.1 route-map policy_google in
```

Se poate observa că în comanda de mai sus pentru vecinul 11.0.1.1 se aplică o politică de tip *out*. Acest parametru îi indică protocolului BGP faptul că *route-map*-ul numit *policy_out* va influența atributele rutelor ce intră în AS-ul propriu.

Se va presupune o topologie în care ruterul R1 este conectat la două IPS-uri diferite și dorește să aleagă mereu ISP-ul identificat prin vecinul BGP 11.0.1.1 pentru traficul spre *google.ro*.

În acest caz, interesant este atributul weight. Acesta poate fi folosit pentru a lua o decizie locală pe un ruter privind direcția în care traficul să se îndrepte în afara AS -ului. Valoarea implicita a atributului pentru orice rută primită este 32768. Setarea unei valori mai mari va face ruta mai atractivă și în același timp preferată în primul pas din procesul de decizie BGP. Configurațiile pentru îndeplinirea scopului descris mai sus pentru traficul ce se îndreaptă spre *google.ro* sunt:

```
R1(config)#route-map policy_google permit 10
R1(config-route-map)#match ip address google_ips
R1(config-route-map)#set weight 40000
R1(config-route-map)#exit
R1(config)#router bgp 100
R1(config-router)#neighbor 11.0.1.1 route-map policy_google in
R1(config-router)#^Z
R1(config)#ip access-list standard google_ips
R1(config-std-nacl)#permit 209.85.229.104 0.0.127.255
R1(config-std-nacl)#permit 74.125.77.104 0.0.255.255
R1(config-std-nacl)#permit 216.239.59.104 0.0.31.255
```

Se creează mai întâi un *route-map* în care se face potrivirea pe IP-urile folosite de *google.ro* printr-o listă de acces standard.

Route-map-ul setează valoarea weight la 40000 pentru a face această rută prioritară altora și apoi acesta este atașat vecinului 11.0.1.1. Astfel toate rutele spre google venite de la acel ISP vor fi preferate și alese ca și ieșire din AS-ul propriu.

Dacă se dorește ca și traficul de întoarcere de la google să vină pe aceeași legătură trebuie manipulat *AS-Path*-ul, însă mai multe despre această operație vor fi descrise în capitolul următor.

8.3.4 Interpretarea tabelelor de topologie BGP

Cea mai importantă comandă din BGP este **`show ip bgp`**. Cu ajutorul acesteia se poate vizualiza tabela BGP de topologie în care sunt prezente *next-hop*-ul pentru fiecare rută alături de câteva din cele mai importante atribute BGP.

```
R2#show ip bgp
BGP table version is 3, local router ID is 150.0.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*>i151.0.1.0/24     150.0.1.1                0    100      0 i
*>i152.0.8.0/22     150.0.1.1                0    100      0 i
```

Ruta cea mai bună este indicată prin folosirea semnului „>". Toate rutele de mai sus au o metrică de 0 și un *local-preference* de 100. Semnificația „i" -ului din stânga este că ruta a fost învățată prin iBGP. Dacă rețeaua ar fi venit de la o rută eBGP acel spațiu dintre „>" și IP-ul rețelei ar fi fost lăsat gol. În partea din dreaptă este listat *AS-PATH*-ul rutei în formă: „AS X AS Y AS Z i/?". Numărul de AS cel mai din dreapta este cel ce a originat ruta. În output-ul de mai sus nu apare niciun AS în listă ceea ce înseamnă că ruta este internă. Ultimul simbol din listă poate fi „i" sau „?", în funcție de modul în care a fost introdusă această rută în BGP: prin comanda **`network`** („i") sau prin comanda **`redistribute`** („?").

8.4 Studiu de caz

Pentru exemplificare vom prezenta o configurație BGP în topologia din figura 3. În acest scenariu, compania noastră (Sky's The Limit) folosește două conexiuni la Internet (*multi-homed AS)* pentru asigurarea redundanței. Ca urmare, compania a decis implementarea BGP pentru a asigura rutarea corectă.

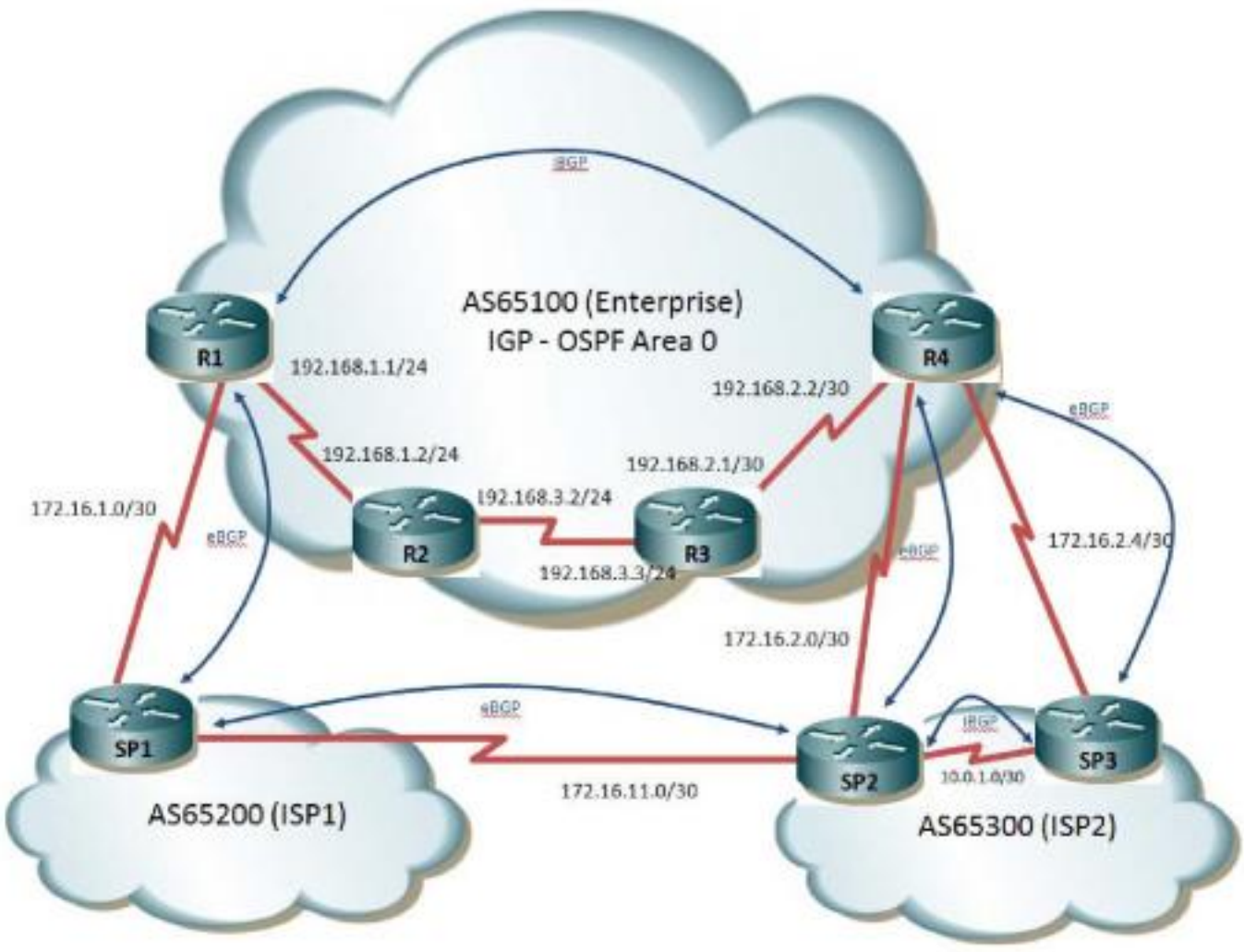

Figura 8-6: : Topologie studiu de caz

Inițial, ambele conexiuni au fost oferite de către același ISP (ISP2), dar acest lucru a dus la apariția unui *single point of failure* în rețeaua companiei - este vorba de R4, a cărui defectare ar putea compromite complet legătura la Internet. Ca urmare, compania a decis achiziționarea unei a treia legături, oferită de un alt ISP (ISP1). Această a treia legătură este plătită în funcție de traficul utilizat și va fi folosită doar atunci când se pierde complet conectivitatea cu ISP2.

8.4.1 Configurații de bază

În rețeaua companiei rulează deja un IGP (OSPF), care anunță rețelele interne (cele aparținând 192.168.0.0/16).

Primul pas în configurarea BGP este pornirea procesului de rutare (cu specificarea AS-ului din care face parte ruterul), și configurarea vecinilor. Pentru fiecare vecin se specifică o adresă IP și AS - ul din care acesta face parte.

De cele mai multe ori, adiacențele eBGP se stabilesc între vecini direct conectați, folosind adresele IP de pe interfețele direct conectate. Dacă IP folosită pentru stabilirea unei adiacențe eBGP nu este direct conectată, este obligatorie folosirea comenzii *neighbor* cu parametrul *ebgp-multihop*.

```
R1(config)#router bgp 65100
R1(config-router)#neighbor 172.16.1.1 remote-as 65200
```

```
SP1(config-if)#router bgp 65200
SP1(config-router)#neighbor 172.16.1.2 remote-as 65100
SP1(config-router)#neighbor 172.16.11.2 remote-as 65200
*Mar  1 00:09:40.075: %BGP-5-ADJCHANGE: neighbor 172.16.1.2 Up
```

```
SP2(config)#router bgp 65300
SP2(config-router)#neighbor 172.16.11.1 remote-as 65200
SP2(config-router)#neighbor 172.16.2.2 remote-as 65100
*Mar  1 00:15:17.595: %BGP-5-ADJCHANGE: neighbor 172.16.11.1 Up
```

```
SP3(config)#router bgp 65300
SP3(config-router)#neighbor 172.16.2.6 remote-as 65100
```

```
R4(config)#router bgp 65100
R4(config-router)#neighbor 172.16.2.1 remote-as 65300
R4(config-router)#neighbor 172.16.2.5 remote-as 65300
*Mar  1 00:16:41.983: %BGP-5-ADJCHANGE: neighbor 172.16.2.1 Up
*Mar  1 00:18:13.351: %BGP-5-ADJCHANGE: neighbor 172.16.2.5 Up
```

Adiacențele iBGP pot fi stabilite între adresele direct conectate (cazul SP2-SP3), sau între adresele altor interfețe, în mod frecvent fiind folosite în acest scop adresele interfețelor de loopback.

Deoarece adresa destinație configurată pe unul dintre rutere trebuie să fie aceeași cu adresa IP sursă folosită de vecinul său, în cazul adiacențelor formate folosind interfețele de loopback devine obligatorie configurarea manuală a adresei IP folosite în update-uri, prin comanda **neighbor** cu parametrul *update-source*.

În exemplul nostru, ruterele R1 și R4 au configurate interfețele Loopback 0 cu adresele IP 192.168.100.1/32, respectiv 192.168.100.4/32.

```
R1(config)#router bgp 65100
R1(config-router)#neighbor 192.168.100.4 remote-as 65100
R1(config-router)#neighbor 192.168.100.4 update-source Loopback 0

R4(config)#router bgp 65100
R4(config-router)#neighbor 192.168.100.1 remote-as 65100
R4(config-router)#neighbor 192.168.100.1 update-source Loopback 0

SP2(config-router)#router bgp 65300
SP2(config-router)#neighbor 10.0.1.2 remote-as 65300

SP3(config-router)#router bgp 65300
SP3(config-router)#neighbor 10.0.1.1 remote-as 65300
*Mar  1 00:28:58.919: %BGP-5-ADJCHANGE: neighbor 10.0.1.1 Up
```

Pentru a verifica starea adiacențelor, se poate folosi comanda **show ip bgp neighbors**, sau **show ip bgp summary**.

```
R4#show ip bgp neighbors
BGP neighbor is 172.16.2.1,  remote AS 65300, external link
  BGP version 4, remote router ID 172.16.11.2
  BGP state = Established, up for 00:13:38
  Last read 00:00:37, last write 00:00:37, hold time is 180, keepalive
interval is 60 seconds
  Neighbor capabilities:
    Route refresh: advertised and received(old & new)
    Address family IPv4 Unicast: advertised and received
  Message statistics:
    InQ depth is 0
    OutQ depth is 0
                         Sent       Rcvd
    Opens:                  1          1
    Notifications:          0          0
    Updates:                0          0
    Keepalives:            15         16
    Route Refresh:          0          0
    Total:                 16         17
  Default minimum time between advertisement runs is 30 seconds
```

```
    R4#show ip bgp summary
    BGP router identifier 192.168.2.2, local AS number 65100
    BGP table version is 1, main routing table version 1

    Neighbor        V    AS MsgRcvd MsgSent   TblVer  InQ OutQ Up/Down
State/PfxRcd
    172.16.2.1      4 65300      19      18        1    0    0 00:15:21        0
    172.16.2.5      4 65300      17      17        1    0    0 00:13:50        0
    192.168.100.1   4 65100       8       8        1    0    0 00:04:07        0
```

În cazul comenzii **show ip bgp summary**, un vecin pentru care conexiunea BGP a fost stabilită cu succes va avea în ultima coloană un număr reprezentând numărul prefixelor primite de la vecinul respectiv. Dacă în coloana respectivă apare un text (*Active* / *Init* / etc), conexiunea nu a fost (încă) stabilită pentru vecinul respectiv.

Următorul pas în configurarea BGP o reprezintă selectarea rutelor care vor fi anunțate. Spre deosebire de IGP-uri, în BGP prefixele anunțate sunt selectate manual, prin comanda **network**. O altă variantă este cea a redistribuirii rutelor din alt protocol în IGP, prin comanda **redistribute**. Diferența dintre cele două abordări este dată de atributul ORIGIN - rutele injectate prin comanda **network** vor avea ORIGIN *internal (i)*, în timp ce rutele redistribuite vor avea ORIGIN *unknown (?)*.

Pentru ca o rută să fie injectată cu succes în BGP, este obligatoriu ca prefixul respectiv să existe în tabela de rutare a ruterului pe care s-a configurat comanda **network**. Dacă prefixul (adresa de rețea sau masca) nu corespund cu o rută existentă, rețeaua nu va fi anunțată.

În exemplul nostru, ruterul R1 va anunța adresele de loopback din interiorul AS 65100 (adresele interfețelor de pe Loopback0 de pe R1 și R4). Ambele adrese există deja în tabela de rutare a lui R1 - adresa proprie este direct conectată, în timp ce adresa R4 este primită prin OSPF. Ca urmare, este posibilă anunțarea respectivelor prefixe în BGP.

În plus, SP1 va anurța prefixul 172.16.11.0/30 (direct conectat), iar SP2 va anunța 10.0.1.0/30 (tot direct conectat).

```
R1(config)#router bgp 65100
R1(config-router)#network 192.168.100.1 mask 255.255.255.255
R1(config-router)#network 192.168.100.4 mask 255.255.255.255
SP1(config)#router bgp 65200
SP1(config-router)#network 172.16.11.0  mask 255.255.255.252


SP2(config)#router bgp 65300
SP2(config-router)#network 10.0.1.0 mask 255.255.255.252
```

Pentru a verifica faptul că rutele sunt transmise corect, vom studia tabela BGP de pe rutere, folosind comanda **`show ip bgp`**. Toate cele patru rute injectate vor trebui să apară pe toate ruterele care vorbesc BGP.

Pentru a obține mai multe detalii despre o rută, se poate folosi varianta **`show ip bgp`** *X.X.X.X* (unde X.X.X.X este adresa de rețea despre care se doresc mai multe detalii).

```
R4#show ip bgp
BGP table version is 7, local router ID is 192.168.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
* i10.0.1.0/30     172.16.1.1               0    100      0 65200 65300 i
*                  172.16.2.1               0             0 65300 i
*>                 172.16.2.5                             0 65300 i
* i172.16.11.0/30  172.16.1.1               0    100      0 65200 i
*>                 172.16.2.1                             0 65300 65200 i
r>i192.168.100.1/32 192.168.100.1          0    100      0 i
r>i192.168.100.4/32 192.168.1.2          193    100      0 i
```

Comanda **`show ip bgp`** este una din cele mai utile comenzi pentru troubleshooting atunci când se folosește BGP. Ea permite vizualizarea tuturor prefixelor cunoscute prin BGP, împreună cu:

- starea acestora (primul simbol de pe fiecare linie);
- cea mai bună cale către o destinație, în cazul în care sunt cunoscute mai multe căi spre aceeași rețea. Cea mai bună cale este marcată cu „>", și va fi cea care va intra în tabela de rutare (pașii urmați pentru alegerea celei mai bune căi au fost descriși într-o secțiune anterioară);
- o listă cu cele mai importante atribute: NEXT_HOP, WEIGHT, LOCAL_PREF, AS_PATH („Path"), ORIGIN (simbolul de la finalul „Path"), MED („Metric").

În exemplul de mai sus, două prefixe (cele aparținând 192.168.0.0/16) sunt marcate ca „r" (*RIB-Failure*). Aceasta nu este o eroare - pur și simplu acestea sunt cunoscute deja de către ruter printr-un alt protocol (192.168.100.4 este direct conectat, iar 192.168.100.1 este primit prin OSPF) și ca urmare nu vor mai fi adăugate în tabela de rutare (*RIB-Routing Information Base*) de către iBGP (care are o distanță administrativă mai mare decât celelalte protocoale).

Pentru a verifica introducerea celor mai bune căi în tabela de rutare, folosim comanda **`show ip route`** (cu varianta **`show ip route bgp`**, pentru a arăta doar rutele cunoscute prin BGP):

```
R4#show ip route bgp
     172.16.0.0/30 is subnetted, 3 subnets
B       172.16.11.0 [20/0] via 172.16.2.1, 00:13:08
     10.0.0.0/30 is subnetted, 1 subnets
B       10.0.1.0 [20/0] via 172.16.2.5, 00:11:51
```

Revenind la tabela BGP de pe R1, observăm o altă problemă menționată în secțiunea „iBGP vs eBGP" - R4 anunță rutele primite de la ISP cu NEXT_HOP nemodificat, și ca urmare R1 nu are o rută către NEXT_HOP-ul respectiv (adresele legăturilor către ISP nu sunt anunțate în OSPF):

```
R1#show ip bgp
BGP table version is 5, local router ID is 192.168.100.1
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 10.0.1.0/30      172.16.1.1                             0 65200 65300 i
* i                 172.16.2.5               0    100      0 65300 i
*> 172.16.11.0/30   172.16.1.1               0             0 65200 i
* i                 172.16.2.1               0    100      0 65300 65200 i
*> 192.168.100.1/32 0.0.0.0                  0         32768 i
*> 192.168.100.4/32 192.168.1.2            193         32768 i

R1#show ip bgp 10.0.1.0
BGP routing table entry for 10.0.1.0/30, version 2
Paths: (2 available, best #1, table Default-IP-Routing-Table)
  Advertised to update-groups:
     2
  65200 65300
    172.16.1.1 from 172.16.1.1 (172.16.11.1)
      Origin IGP, localpref 100, valid, external, best
  65300
    172.16.2.5 (inaccessible) from 192.168.100.4 (192.168.2.2)
      Origin IGP, metric 0, localpref 100, valid, internal
```

Cea mai simplă soluție pentru această problemă este configurarea R1 (și a oricărui alt ruter care primește rute prin eBGP și le anunță mai departe în iBGP) pentru a modifica NEXT_HOP. Comanda folosită este **`neighbor ... next-hop-self`** (dată pentru fiecare vecin iBGP al ruterului respectiv).

```
R1(config)#router bgp 65100
R1(config-router)#neighbor 192.168.100.4 next-hop-self
```

```
R4(config)#router bgp 65100
R4(config-router)#neighbor 192.168.100.1 next-hop-self
```

Rezultatul este că rutele sunt acum primite cu un NEXT_HOP care este intern (din AS 65100),și cunoscut prin OSPF.

```
R1#show ip bgp
BGP table version is 6, local router ID is 192.168.100.1
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*  10.0.1.0/30      172.16.1.1                             0 65200 65300 i
*>i                 192.168.100.4            0    100      0 65300 i
*> 172.16.11.0/30   172.16.1.1               0             0 65200 i
*> 192.168.100.1/32 0.0.0.0                  0         32768 i
*> 192.168.100.4/32 192.168.1.2            193         32768 i
```

Aceeași configurație este necesară și pentru SP2 și SP3:

```
SP2(config-router)#router bgp 65300
SP2(config-router)#neighbor 10.0.1.2 next-hop-self
```

```
SP3(config-router)#router bgp 65300
SP3(config-router)#neighbor 10.0.1.1 next-hop-self
```

În acest moment, configurația de bază pare completă. Tabelele de rutare sunt corect populate (prin OSPF, respectiv BGP),și complete. Totuși, mai există o problemă importantă, care afectează comunicația prin rețea:

```
R1#show ip route

     172.16.0.0/30 is subnetted, 2 subnets
B       172.16.11.0 [20/0] via 172.16.1.1, 00:19:32
C       172.16.1.0 is directly connected, Serial0/1
     10.0.0.0/30 is subnetted, 1 subnets
B       10.0.1.0 [200/0] via 192.168.100.4, 00:12:42
C    192.168.1.0/24 is directly connected, Serial0/0
     192.168.2.0/30 is subnetted, 1 subnets
O       192.168.2.0 [110/192] via 192.168.1.2, 00:54:03, Serial0/0
     192.168.100.0/32 is subnetted, 2 subnets
O       192.168.100.4 [110/193] via 192.168.1.2, 00:54:03, Serial0/0
C       192.168.100.1 is directly connected, Loopback0
O    192.168.3.0/24 [110/128] via 192.168.1.2, 00:54:03, Serial0/0

R1#ping 10.0.1.1 source Lo0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.0.1.1, timeout is 2 seconds:
Packet sent with a source address of 192.168.100.1
U.U.U
Success rate is 0 percent (0/5)
```

Ruta există, dar pachetele de date trimise către destinația respectivă se pierd. Este vorba din nou de o problemă specifică BGP, provenită din faptul că adiacențele BGP se pot realiza între rutere care nu sunt direct conectate fizic.

În cazul nostru, adiacența iBGP dintre R1 și R4 „sare peste" R2 și R3. În schimb, pachetele de date trimise de la R1 către R4 vor trebui însă să treacă prin R2 și R3. În momentul în care pachetul de ping cu destinație 10.0.1.1 ajunge la R2, acesta nu cunoaște o rută către destinația respectivă (R2 nu vorbește BGP, și ca urmare nu va avea rute către destinațiile din afara AS-ului propriu). Ca urmare, toate pachetele de date cu respectiva destinație vor fi *discarded*.

Există două soluții principale pentru această problemă, fiecare soluție având o serie de dezavantaje:

- Redistribuirea rutelor primite prin BGP în IGP-ul folosit (în cazul nostru redistribuirea din BGP în OSPF). Soluția are dezavantajul lipsei de scalabilitate - practic nici un IGP nu va putea să proceseze numărul imens de rute transportate de BGP (peste 290.000 de prefixe în 2009);
- Rularea BGP pe toate ruterele din sistemul autonom. De multe ori singura soluție posibilă (mai ales în cazul AS-urilor de tranzit). Dezavantajul ei este dat de faptul că iBGP necesită crearea unui *full-mesh* între toate ruterele din AS. Ca urmare, numărul de adiacențe iBGP necesare va crește exponențial cu numărul de rutere din sistemul autonom.

Deoarece în exemplul nostru se primesc doar două rute din afara AS, vom alege prima variantă - redistribuirea rutelor în IGP.

```
R4(config)#router ospf 1
R4(config-router)#redi
R4(config-router)#redistribute bgp 65100 sub
R4(config-router)#redistribute bgp 65100 subnets
```

Rezultat - acum traficul traversează în mod corect R2 și R3:

```
R1#ping 10.0.1.1 source Lo0

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.0.1.1, timeout is 2 seconds:
Packet sent with a source address of 192.168.100.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 40/67/120 ms
```

8.4.2 Manipularea atributelor BGP

Configurările de bază fiind realizate, putem trece la implementarea politicii de rutare din AS 65100. Implementarea realizată va trebui să satisfacă următoarele cerințe:

- legătura R4-SP3 este legătura principală la Internet a AS65100;
- legătura R4-SP2 este legătura secundară (folosită doar atunci când legătura principală cade);
- legătura R1-SP1 va fi folosită doar atunci când se pierd ambele legături de pe R4.

Pentru o implementare corectă a politicii, trebuie ca legăturile să fie folosite în acest mod atât de traficul care părăsește AS -ul, cât și de traficul de intrare. Acest lucru presupune modificarea deciziei unor rutere care aparțin altor AS-uri (AS 65200, AS 65300) prin setări realizate doar în AS-ul propriu (AS 65100).

În BGP, politica de rutare se implementează prin modificarea atributelor rutelor primite, respectiv trimise prin BGP. În acest scop se folosesc *route-maps* (descrise în capitolul de *route optimization*).

Pentru a afecta traficul de ieșire se poate modifica atributul LOCAL_PREFERENCE. Ne vom folosi de faptul că valoarea implicită a acestuia este 100, și vom seta noile valori astfel:

- pentru rutele primite de la SP2, LOCAL_PREF=50;
- pentru rutele primite de la SP1, LOCAL_PREF=30.

```
R4(config)#route-map LP_50 permit 10
R4(config-route-map)#set local-preference 50
R4(config-route-map)#exit

R4(config)#router bgp 65100
R4(config-router)#neighbor 172.16.2.1 route-map LP_50 in

!! Orice modificare a politicii de rutare necesită reinițializarea
!! adiacenței BGP!

R4#clear ip bgp *
```

```
R1(config)#route-map LP_30 permit 10
R1(config-route-map)#set local-preference 30

R1(config-route-map)#router bgp 65100
R1(config-router)#neighbor 172.16.1.1 route-map LP_30 in
R1(config-router)#^Z

R1#clear ip bgp *
```

Rezultatul îl putem observa atât în tabela BGP, cât și în tabela de rutare. De exemplu, în cazul R4, calea preferată de ieșire din AS va fi prin SP3:

```
R4#show ip bgp
BGP table version is 14, local router ID is 192.168.2.2
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*  10.0.1.0/30      172.16.2.1               0     50      0 65300 i
*>                  172.16.2.5                             0 65300 i
*  172.16.11.0/30   172.16.2.1                50           0 65300 65200 i
*>                  172.16.2.5                             0 65300 65200 i
r>i192.168.100.1/32 192.168.100.1            0    100      0 i
r>i192.168.100.4/32 192.168.100.1          193    100      0 i

R4#show ip route bgp
     172.16.0.0/30 is subnetted, 3 subnets
B       172.16.11.0 [20/0] via 172.16.2.5, 00:04:44
     10.0.0.0/30 is subnetted, 1 subnets
B       10.0.1.0 [20/0] via 172.16.2.5, 00:04:57
```

O problemă mai dificilă este reprezentată de modificarea căii urmate de traficul de intrare în AS. În cazul SP2 și SP3 (rutere care fac parte din același AS - 65200), problema poate fi rezolvată prin modificarea MED. Valorile mai mici pentru MED sunt preferateși ca urmare vom seta MED la 50 pentru legătura R4-SP3, și la 100 pentru R4-SP2.

Setările vor fi realizate pentru rutele care sunt trimise către vecinii din alte AS-uri (*outbound route-maps*).

```
R4(config)#route-map MED_50 permit 10
R4(config-route-map)#set metric 50
R4(config-route-map)#exit

R4(config)#route-map MED_100 permit 10
R4(config-route-map)#set metric 100
R4(config-route-map)#exit

R4(config)#router bgp 65100
R4(config-router)#neighbor 172.16.2.5 route-map MED_50 out
R4(config-router)#neighbor 172.16.2.1 route-map MED_100 out
R4(config-router)#^Z

R4#clear ip bgp *
```

Rezultat - SP2 va trimite pachetele destinate AS100 via SP3:

```
SP2#show ip bgp
BGP table version is 30, local router ID is 172.16.11.2
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*> 10.0.1.0/30      0.0.0.0                  0         32768 i
r> 172.16.11.0/30   172.16.11.1              0             0 65200 i
*>i192.168.100.1/32 10.0.1.2                50    100      0 65100 i
*                   172.16.2.2             100             0 65100 i
*                   172.16.11.1                            0 65200 65100 i
*>i192.168.100.4/32 10.0.1.2                50    100      0 65100 i
*                   172.16.2.2             100             0 65100 i
*                   172.16.11.1                            0 65200 65100 i
SP2#show ip route bgp
     192.168.100.0/32 is subnetted, 2 subnets
B       192.168.100.4 [200/50] via 10.0.1.2, 00:00:48
B       192.168.100.1 [200/50] via 10.0.1.2, 00:00:48
```

Problema principală a MED este faptul că acesta nu este transmis la următoarele sisteme autonome. Ca urmare, nu este posibilă modificarea deciziei AS 65200 (sau a altor AS-uri, aflate mai departe) folosind doar MED.

În aceste cazuri, soluția preferată este modificarea AS_PATH. Acesta este modificat prin adăugarea la începutul său (*prepending)* a ASN-ului propriu, repetat de un număr de ori. Cu cât numărul de repetiții va fi mai mare, cu atât ruta respectivă va fi considerată mai puțin bună de către ruterele din alte AS-uri. În general (în Internet), este suficient un *AS-Path prepending* de 3 sau 4 ori.

Este obligatoriu ca ASN-ul folosit pentru prepending să fie ASN-ul propriu!

```
R1(config)#route-map PREPEND permit 10
R1(config-route-map)#set as-path prepend 65100 65100 65100
R1(config-route-map)#router bgp 65100
R1(config-router)#neighbor 172.16.1.1 route-map PREPEND out
R1(config-router)#^Z
R1#clear ip bgp *
```

Rezultatul configurației se observă pe SP1 - ruta prin R1 are un AS_PATH mai lung decât ruta alternativă (prin SP2), și ca urmare SP1 va trimite pachetele destinate AS 65100 către SP2:

```
SP1#show ip bgp
BGP table version is 25, local router ID is 172.16.11.1
Status codes: s suppressed, d damped, h history, * valid, > best, i -
internal,
              r RIB-failure, S Stale
Origin codes: i - IGP, e - EGP, ? - incomplete

   Network          Next Hop            Metric LocPrf Weight Path
*  10.0.1.0/30      172.16.1.2                             0 65100 65100
65100 65100 65300 i
*>                  172.16.11.2              0             0 65300 i
*> 172.16.11.0/30   0.0.0.0                  0         32768 i

SP1#show ip route bgp
     10.0.0.0/30 is subnetted, 1 subnets
B       10.0.1.0 [20/0] via 172.16.11.2, 01:54:03
     192.168.100.0/32 is subnetted, 2 subnets
B       192.168.100.4 [20/0] via 172.16.11.2, 00:00:51
B       192.168.100.1 [20/0] via 172.16.11.2, 00:00:51
```

Trebuie menționat aici faptul că modificarea traseului traficului de intrare se putea face numai prin *AS_PATH prepending* (de exemplu prin *prepending* de 2 ori pe SP3 și de 3 ori pe SP1), și aceasta este în general soluția preferată. Varianta folosirii MED a fost prezentată doar ca exemplu didactic.

9 IP Multicast

Conceptul de multicast s-a născut la sfârșitul anilor 1980 când Steve Deering a lucrat la un proiect ce presupunea transmiterea unui mesaj de la un calculator la un grup de calculatoare peste o infrastructura de nivel 3. Analizând protocoalele de rutare existente în acel moment, Deering a tras concluzia că acestea pot fi extinse pentru a implementa ceea ce el a numit *„Layer 3 Multicasting"*. În urma unei perioade de cercetare a fost publicată lucrarea *„Multicast Routing in a Datagram Network"* în care erau definite componentele principale ale unei infrastructuri multicast, protocoalele multicast și funcționalitățile lor. În capitolul ce urmează se vor analiza cerințele unei rețele multicast moderne și protocoale ce au evoluat din 1991, de la publicarea lucrării mai sus menționate, până în prezent.

Plecând de la conceptul de bază a lui Dr. Deering, infrastructurile din prezent pot lua forme extrem de complexe însă cea mai simplă definiție a conceptului de IP multicast este:

Trimiterea unui mesaj de la o singură sursă la mai multe destinații selectate folosind un singur flux de date peste o rețea de nivel 3.

9.1 De ce avem nevoie de multicast ?

Pentru a răspunde la această întrebare se vor analiza alternativele pe care protocolul IP le oferă. Dacă se dorește trimiterea unui mesaj de la o singură sursă la o singură destinație, se poate folosi un mesaj **unicast**. Dacă se dorește trimiterea unui mesaj de la o singură sursă la toate destinațiile dintr-o subrețea, se va folosi un mesaj **broadcast**. Însă dacă se dorește trimiterea unui mesaj de la o sursă (sau chiar mai multe) la multiple destinații într-un singur *stream* de date, peste o rețea de nivel 3, cel mai eficient mecanism este **IP multicast**.

Ca și popularitate, transmiterea de multicast este folosită din ce în ce mai mult în rețele moderne pentru *streaming* de video, jocuri *multiplayer*, transmiterea de radio etc. De asemenea în IPv6 multicast joaca un rol important deoarece transmisiile de tip *broadcast* nu mai sunt implementate.

9.2 Funcționarea multicast

In cele ce urmează se va realiza un studiu de caz pentru a motiva existența tipului multicast de transmisie de date, față de folosirea unicast sau broadcast pentru a îndeplini același scop.

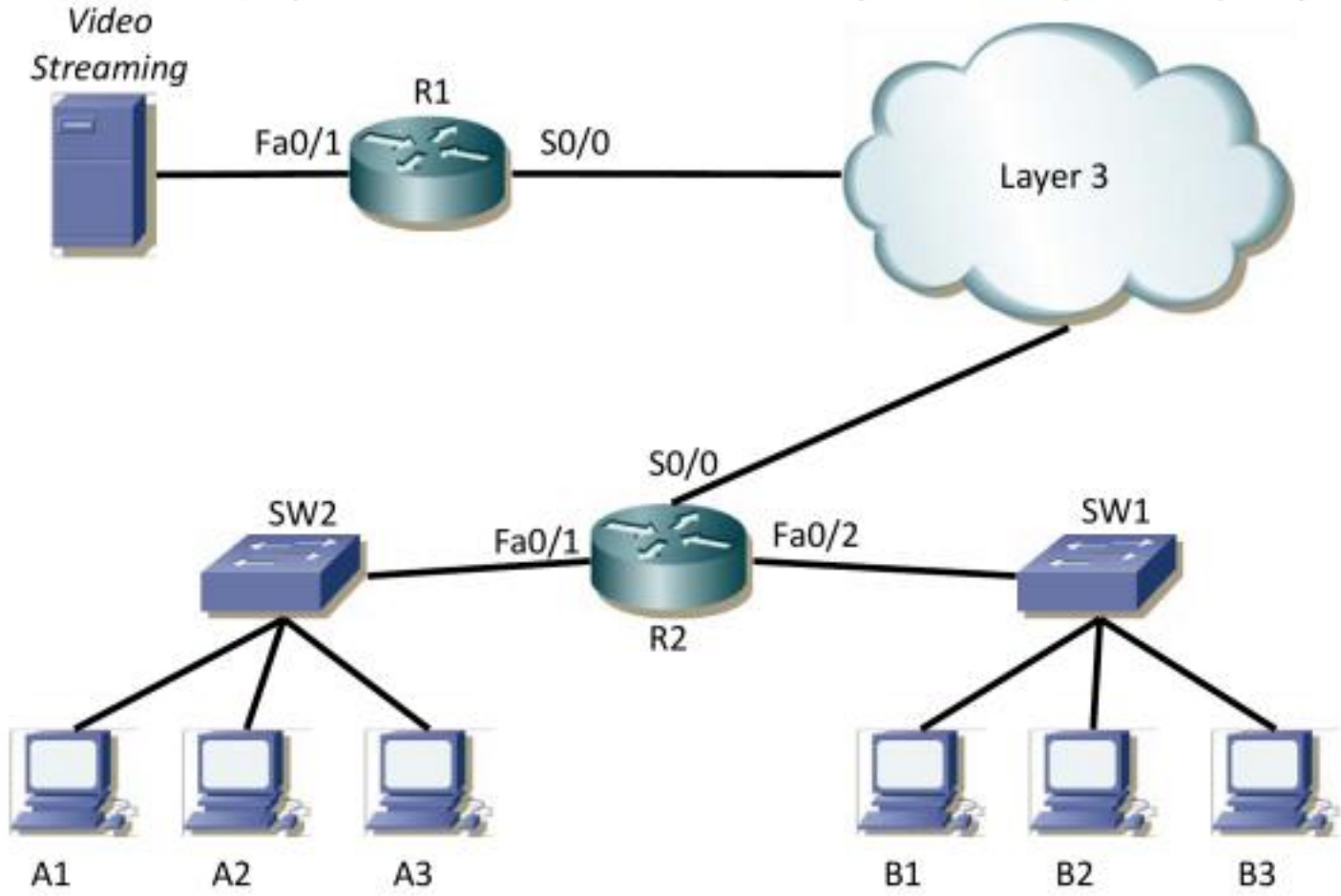

Figura 9-1: Infrastructură IP

În topologia de mai sus se dorește transmiterea unui *stream* video de la Serverul direct conectat la ruterul R1, peste Internet, la multiple stații destinație conectate la SW1 și SW2.

Cazul 1: unicast

Paradigma *unicast* folosește o singură sursă și destinație. Plecând de la această definiție, pentru a putea atinge scopul definit mai sus, serverul de *Video Streaming* va trebui să transmită **n** *stream*-uri diferite. Dacă se notează valoarea lățimii de bandă necesară pentru a transmite *stream*-ul video cu b, rezultă o bandă totală de n*b ce trebuie consumată pe întreaga rută de sursă la destinație. Se concluzionează faptul că această soluție este ineficientă și greu scalabilă.

Cazul 2: broadcast

Față de *unicast*, *broadcast*-ul are avantajul transmiterii unui singur stream. Problemele apar însă din modul în care dispozitivele de rețea tratează *broadcast*-urile. Un ruter nu poate să trimită pachete *broadcast* în mod implicit dintr-o rețea de nivel 3 în alta, iar odată configurat acest comportament, pachetele se vor duce în toate direțiile chiar dacă destinațiile pentru acel stream sunt doar câteva stații izolate într-o rețea locală. Această problemă apare ca o consecință a faptului că o adresă de broadcast se identifică cu o rețea locală, nu cu unul sau mai multe grupuri selectate de multicast.

Cazul 3: multicast

Pentru a putea transmite multicast este nevoie atât de infrastructura de nivel 3 multicast, cât și de suport din partea nivelului aplicație. Cu alte cuvinte atât la sursă cât și la destinație este instalată aplicația de multicast care să trimită pachete cu adresa destinație adresa grupului de multicast și care să știe să primească și să interpreteze aceste pachete la destinație. Transmisia multicast are nevoie de următoarele componente pentru a funcționa corect:

- **O clasă specială de adrese** din care se vor delega adrese pentru fiecare grup multicast. Aceasta este identificată ca fiind clasa D din protocolul IP: 224.0.0.0 – 239.255.255.255. Adresele clasei D pot fi ușor recunoscute prin primii 4 biți ai primului octet: 1110;
- Fiecare **grup** trebuie să fie identificat de o **adresă din spațiul clasei D**. O astfel de adresă nu poate fi folosită decât ca adresă destinație, niciodată ca adresă sursă. Față de IP unicast, multicast nu este folosit pentru a identifica stații sau echipamente, ci doar grupuri destinație;
- **Aplicația multicast** trebuie instalată pe toate stațiile ce doresc să primească *stream*-ul de la server. De asemenea, aceasta trebuie configurată să asculte pe acea adresă IP pe care serverul trimite traficul;
- **O metodă standard de a calcula o adresă de nivel 2 dintr-o adresă de multicast de nivel 3.** Pentru că în stiva TCP/IP se verifică mai întâi adresa MAC, trebuie să existe o asociere între o adresă multicast de nivel 3 și adresa multicast de nivel 2. Cu alte cuvinte, când o aplicație client de multicast pornește și este configurată cu adresa IP a grupului de multicast, ea va deduce în mod standard o adresa MAC din adresa IP și va configura placa de rețea să asculte pe acea adresă MAC pentru traficul de la server.
- **Un mecanism prin care stațiile dintr -o rețea locală să poată semnala ruterului gateway faptul că doresc să primească trafic pentru un grup semnalat**. Totodată, trebuie să existe o metodă prin care o stație să poată părăsi un grup de multicast. Aceste funcționalități sunt oferite prin intermediul protocolului *Internet Group Management Protocol* (IGMP).
- **Un protocol de rutare multicast** care să transmită *stream*-ul peste infrastructura de nivel 3. Cele mai cunoscute protocoale ce îndeplinesc acest scop sunt: *Distance Vector Multicast Routing Protocol* (DVMRP), *Protocol Independent Multicast* (PIM), *Multicast OSPF* (MOSPF).

Urmărind pașii de mai jos se va oferi o imagine de ansamblu asupra funcționării unei transmisii multicast în Figura 1-1 de la sursă la destinație. În capitolele ulterioare se va pune accept pe implementarea fiecărui protocol menționat.

1. Serverul video va transmite un singur *stream* de trafic multicast către adresa specială multicast 224.0.1.120, ce identifică grupul din care fac parte stațiile A1, A2 și B2.
2. Pentru direcționarea *stream*-ului de la ruterul R1 la R2 peste Internet, se va folosi un protocol de rutare multicast care va transmite traficul doar acelor rutere care fac parte din cea mai scurta cale de la R1 la R2.
3. Stațiile A1, A2 și B2 vor folosi protocolul IGMP pentru a îi specifica ruterul R2 că doresc să primească trafic pentru grupul multicast 224.0.1.120. Ruterul va lua deci decizia direcționării traficului multicast primit către interfețele Fa0/1 și Fa0/2. Pentru a trimite însă trafic într-o rețea locală, R2 trebuie să elimine antetul de nivel 2 de pe legătura serială S0/0 și să încapsuleze cu un antet Ethernet (o adresă MAC sursă și destinație). Traficul multicast nu are ca destinație o singură stație (adresă MAC unicast) sau toate stațiile (adresa MAC FFFF:FFFF:FFFF), astfel încât ruterul va utiliza adresa MAC destinație: 0x0100.5e00.0178 din adresa de nivel 3: 224.0.1.120.
4. Odată instalate, pentru a primi trafic pentru grupul 224.0.1.120, aplicația client de pe stațiile A1, A2, B2 vor configura placa de rețea pentru a asculta pe adresa MAC 0x0100.5e00.0178.
5. Switchurile Sw1 și Sw2 vor primi traficul multicast de la R2 și, pentru că în mod implicit nu au adresa 0x0100.5e00.0178 în tabela CAM, vor face *flooding*, trimițând traficul la toate stațiile din LAN. Astfel traficul va ajunge atât la stațiile care au cerut prin IGMP acest *stream* cât și la stațiile care nu doreau să facă parte din grupul multicast 224.0.1.120 și care vor ignora pachetele primite la nivel 2 (plăcile lor de rețea nu ascultă pe adresa 0x0100.5e00.0178). Acest comportament ineficient poate fi împiedicat folosind protocoale precum Cisco Group Management Protocol (CGMP) sau IGMP Snooping care pot indica switchului pe ce porturi se află stațiile care doresc trafic multicast pentru un anumit grup.

6. R2 va continua să trimită trafic pe ambele interfețe FastEthernet până când *stream*-ul se va termina sau până când toate stațiile de pe oricare dintre segmentele LAN vor părăsi grupul de multicast 224.0.1.120 folosind un mesaj de IGMP *Leave*.

Un ultim detaliu de menționat în această introducere este că traficul multicast funcționează numai peste UDP.

9.3 Adresarea multicast

Existența unui spațiu de adrese IP alocat transmisiilor multicast este vitală în funcționarea conceptului de IP Multicast. În paradigma unicast, o adresă IP reprezenta un nod și de aceea se realiza și asignarea acesteia pe o interfață a respectivului nod. În contrast, o adresă multicast reprezintă un grup de destinatari și nu este niciodată asignată unui dispozitiv. Singurul loc în care acest tip de adresă este folosită este în câmpul de IP destinație din antetul IP pentru ca dispozitivele de rutare să recunoască acel pachet ca făcând parte dintr-un *stream* de la sursa: IP_SURSĂ și destinat grupului: IP_DESTINAȚIE.

Organizația IANA (*Internet Assigned Numbers Authority*) a alocat întreaga clasa D pentru utilizarea în aplicații multicast: 224.0.0.0 – 239.255.255.255. Primii patru biți sunt mereu 1110 iar ultimii 28 de biți reprezintă o adresare plată (o adresă multicast nu are o mască de rețea).

9.3.1 Structura adreselor multicast

Internet Assigned Numbers Authority (IANA) este organizația ce controlează asignarea adreselor multicast către noi aplicații emergente. Pentru a nu rămâne fără adrese IP, IANA este reticentă la oferirea adreselor din clasa D. O aplicație trebuie să fie nu numai stabilă şi bine construită, ci să demonstreze și o acoperire considerabilă a piței. Adresele multicast se împart în următoarele categorii:

Adresele permanente

Adrese multicast permanente se alocă din spațiul de adrese 224.0.0.0 – 224.0.1.255. Aceste adrese sunt asociate de către IANA protocoalelor şi aplicațiilor bine cunoscute. Ele se împart în două subtipuri funcție de criteriul: dacă sunt direcționate de ruter sau nu.

Spațiul de adrese 224.0.0.0 – 224.0.0.255 sunt cunoscute ca fiind adrese multicast locale pentru că nu sunt direcționate niciodată de către un ruter. În această categorie intră adresele folosite de diferitele protocoale de rutare:

- EIGRP: 224.0.0.9
- RIPv2: 224.0.0.10
- OSPF: 224.0.0.5 şi 224.0.0.6

Restul spațiului (224.0.1.0 – 224.0.1.255) este alocat celor mai cunoscute aplicații multicast şi sunt direcționate de către rutere peste rețele de nivel 3:

- 224.0.1.1 NTP
- 224.0.1.75 SIP

Adresele private

Echivalentul din unicast al adreselor private, adresele multicast din spațiul 239.0.0.0 – 239.255.255.255 este folosit în domenii private de administrare, fiind responsabilitatea administratorului de rețea (și a ISP -urilor, în ultimă instanță) să nu permită direcționarea pachetelor ce conțin astfel de adrese într-o rețea publică.

Adresele Source Specific Multicast

Adresele SSM se alocă din spațiul de adrese 232.0.0.0 – 232.255.255.255 şi sunt folosite în cazuri în care destinația doreşte să aibă posibilitatea de a selecta sursa de la care doreşte să

primească trafic. Acest mecanism poate fi folosi şi pentru a limita încercările de DoS. Dacă una dintre sursele traficului de multicast devine compromisă şi începe să trimită trafic nedorit, utilizatorul poate comunica protocolului faptul că nu doreşte să primească trafic de la respectiva sursă.

Adresele GLOP

Spațiul de adrese pentru acest tip special de multicast este: 233.0.0.0 – 233.255.255.255. Ideea adreselor GLOP a fost concepută pentru a da posibilitatea fiecărui deținător al unui AS să poată folosi 255 de adrese multicast proprii, unice la nivel global. Implementarea este foarte simplă: primii 8 biți ai unei adrese GLOP sunt mereu „233", după care urmează cei 16 biți ai numărului AS şi apoi 8 biți pe care fiecare organizație îi poate varia pentru a obține 255 de adrese multicast, rutabile la nivel global. În mod implicit, IANA permite existența acestor adrese de multicast în Internet fără a fi nevoie de o procedură de închiriere sau mapare a adresei.

Adresele „transient"

Restul spațiului multicast este format din adrese rutabile la nivel global, însă temporare. Dacă o organizație doreşte să închirieze o adresă multicast pentru o anumită perioadă/transmisie, IANA oferă o procedură de *lease/unlease* care face posibil acest lucru.

9.3.2 Maparea adreselor multicast IP la adrese MAC

După cum s-a specificat în capitolul 1.2, adresarea multicast se face atât la nivel 3 cât şi la nivel 2, cea din urmă fiind dedusă printr-o formulă din cea dintâi. Configurarea unei aplicații pentru utilizarea unei adrese IP multicast generează automat o adresă MAC pe care placa de rețea va asculta pentru cadre ce fac parte din *stream*. Construirea unei astfel de adrese se face astfel:

- Partea OUI a adresei MAC este mereu: 01005E;
- Bitul 24 este mereu 0
- Ultimii 23 de biți ai adresei MAC sunt identici cu ultimii 23 de biți ai adresei IP.

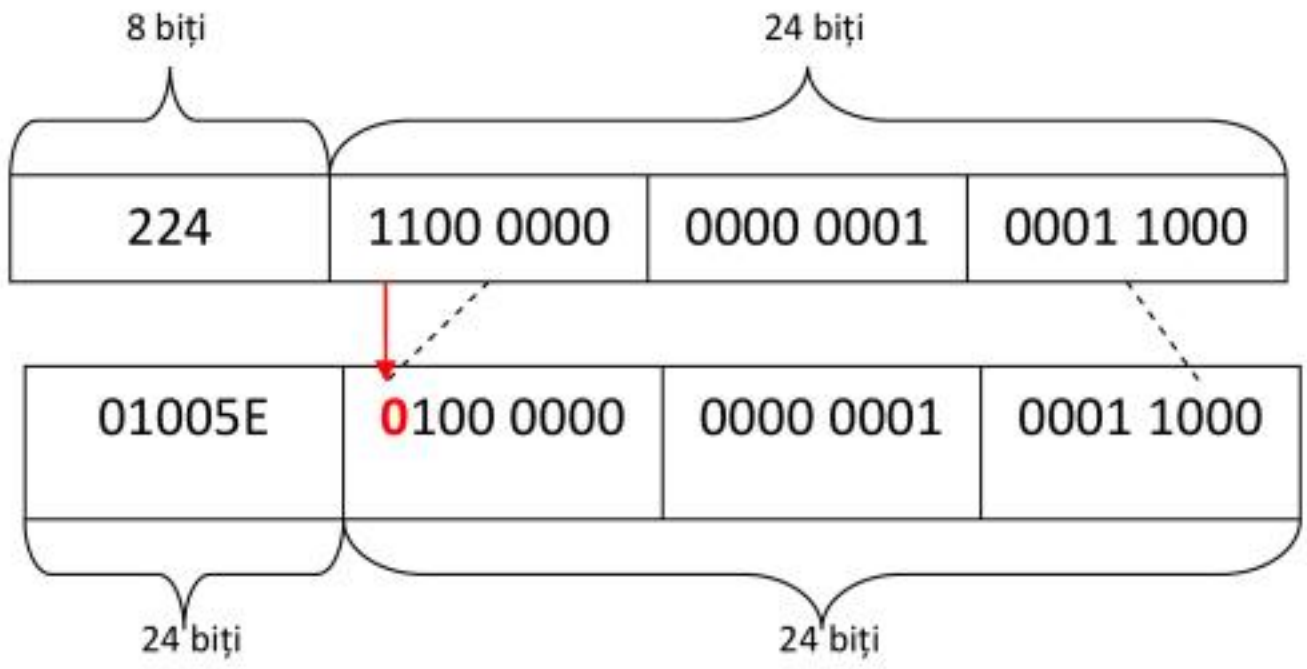

Figura 9-2: Maparea dintre o adresă multicast şi o adresă MAC

Pentru a exemplifica acest procedeu a fost folosită adresa multicast 224.192.1.24. Primii 8 biți ai adresei IP sunt scrişi în zecimal şi primii 24 de biți ai adresei MAC sunt scrişi în hexazecimal din motive de ergonomie.

9.4 Gestiunea grupurilor multicast în LAN

În cele de mai jos se va reveni asupra scenariului prezentat în capitolul 1.2 Funcționarea multicast. Se va considera momentul din transmisie în care traficul multicast a ajuns prin seriala S0/0 la R2. Ruterul va trebui să poată să răspundă la următoarele întrebări:

- Este cineva pe orice segment LAN direct conectat care dorește acest *stream* video ?
- Dacă există cel puțin un destinatar al *stream*-ului pe care dintre interfețele Ethernet ar trebui să fie transmis traficul multicast ?

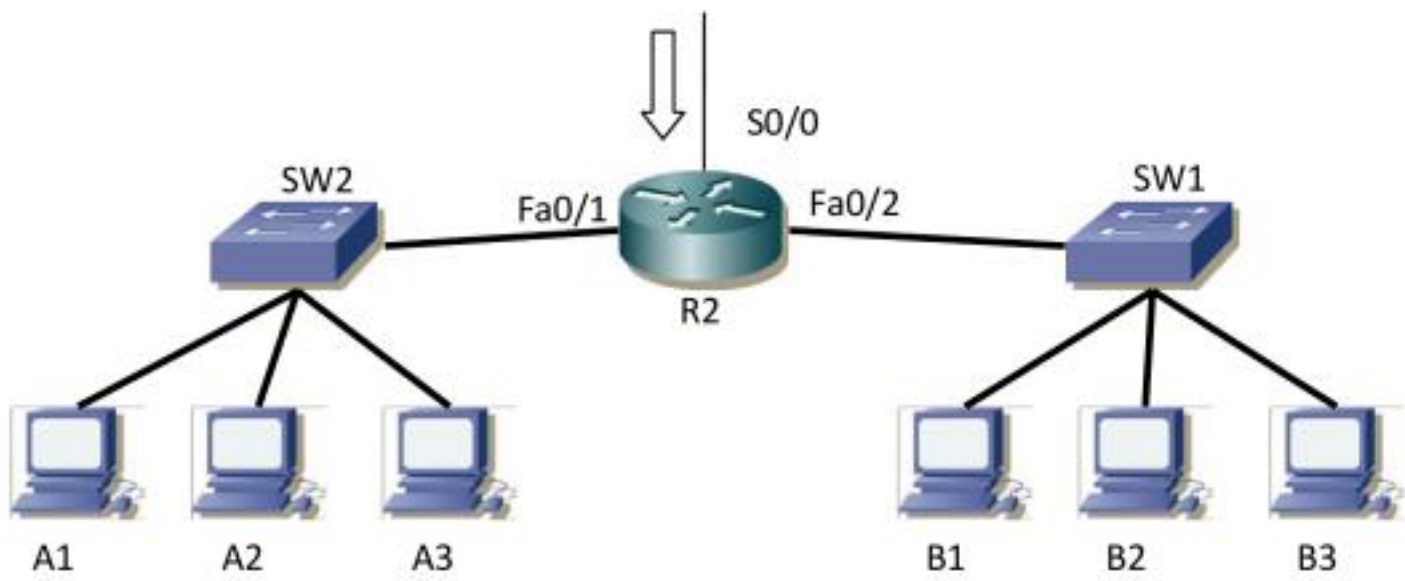

Figura 9-3: Managementul traficului în LAN

Întrebările de mai sus definesc, de fapt, funcțiile de bază pe care protocolul IGMPv2 le implementează. În topologia de mai sus, însă, nu doar ruterul are responsabilitatea de decizie. Implementarea unei soluții multicast complete presupune ca switchurile din figură să cunoască porturile pe care se află stațiile care fac parte din grupul multicast și să trimită *stream*-ul de la ruter doar pe acele porturi. Pentru a avea un astfel de comportament, un switch trebuie să ruleze unul din protocoalele*: Cisco Group Management Protocol* (CGMP) sau *IGMP Snooping*.

9.4.1 IGMP v2

IGMP este protocolul ce a evoluat din descrierea lui Deering. Deși în prezent protocolul a ajuns la versiunea a 3-a, această carte va trata versiunea a 2-a căci este cea folosită în 99% dintre cazuri în implementările din industrie. IGMPv3 nu este suportat deocamdată pe niciun sistem de operare și suportul din partea aplicațiilor este și el foarte limitat. În contrast, IGMPv1 este o versiune veche și ineficientă a protocolului care va fi menționată doar în conturarea evoluției facilităților din IGMPv2.

Rolurile de bază ale protocolului IGMP sunt următoarele:

- Oferirea unui mecanism prin care o stație să poată informa ruterul *gateway* de faptul că dorește să primească trafic multicast pentru un anumit grup;
- Oferirea unui mecanism prin care o stație să poată informa ruterul *gateway* că dorește să părăsească un grup;

IGMPv2 funcționează direct peste protocolul IP (următorul antet după IP), având numărul de protocol 2. Toate mesajele folosite în protocol au câmpul TTL egal cu 1 pentru a limita raza acestora la rețeaua locală.

9.4.2 Antetul IGMPv2

Pentru a explica mecanismul de înregistrare la un grup de multicast, se va prezenta mai întâi antetul IGMPv2:

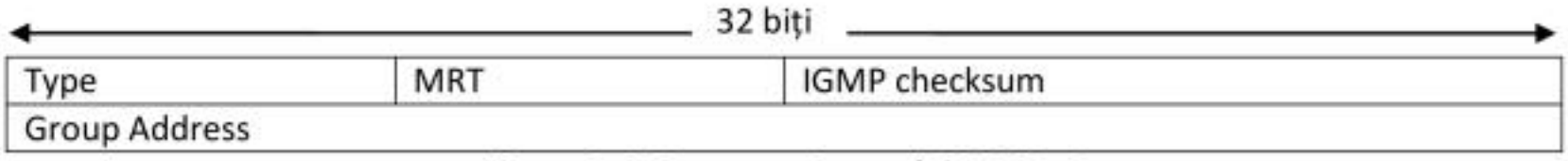

Figura 9-4: Structura Antetului IGMPv2

Câmpul *Type* este folosit pentru codificarea diferitelor tipuri de mesaje IGMP. Se amintesc:

- 0x11 – **Membership Query** este mesajul folosit de ruter pentru a interoga existența destinatarilor pentru un grup de multicast pe un segment de rețea. Există două tipuri de *query*-uri folosite în IGMPv2: **General Membership Query** și **Group-Specific Query**.
- 0x16 – **Membership Report** este folosit de stații pentru a semnala ruterului că cel puțin un destinatar al unui grup specific de multicast există pe segmentul de rețea.
- 0x17 – **Leave Group** este folosit pentru a semnala ruterului *gateway* faptul că un destinatar nu mai dorește să primească trafic multicast.

Câmpul MRT (*Maximum Response Time*) are o valoare între 0.1 și 25.5 secunde și este folosit pentru optimizarea procesului de înregistrare la un grup și se va detalia în acest subcapitol. Acesta este exprimat în antet în zecimi de secunde: o valoare de 30 va însemna 3 secunde. În mod implicit, valoarea este de 10 secunde într-un General *Membership Query*.

Câmpul IGMP *checksum* este folosit ca și orice alt *checksum* din protocoale de comunicație pentru a verifica integritatea mesajului.

Câmpul *Group Address* este poate cel mai semnificativ din antet în înțelegerea funcționalității de înregistrare IGMP și este setat la diferite valori funcție de mesajul folosit. Spre exemplu, un General *Membership Query* va seta acest câmp la valoarea 0.0.0.0 căci dorește să descopere orice destinatar al oricărui grup multicast pe un anumit segment.

9.4.3 Înregistrarea la un grup de multicast

Cele 2 mesaje folosite în înregistrarea la un grup multicast sunt: *General Membership Query* și *Membership Report*. În continuare se vor prezenta pașii din procesul de trimitere a primului dintre aceste două mesaje:

General Membership Query

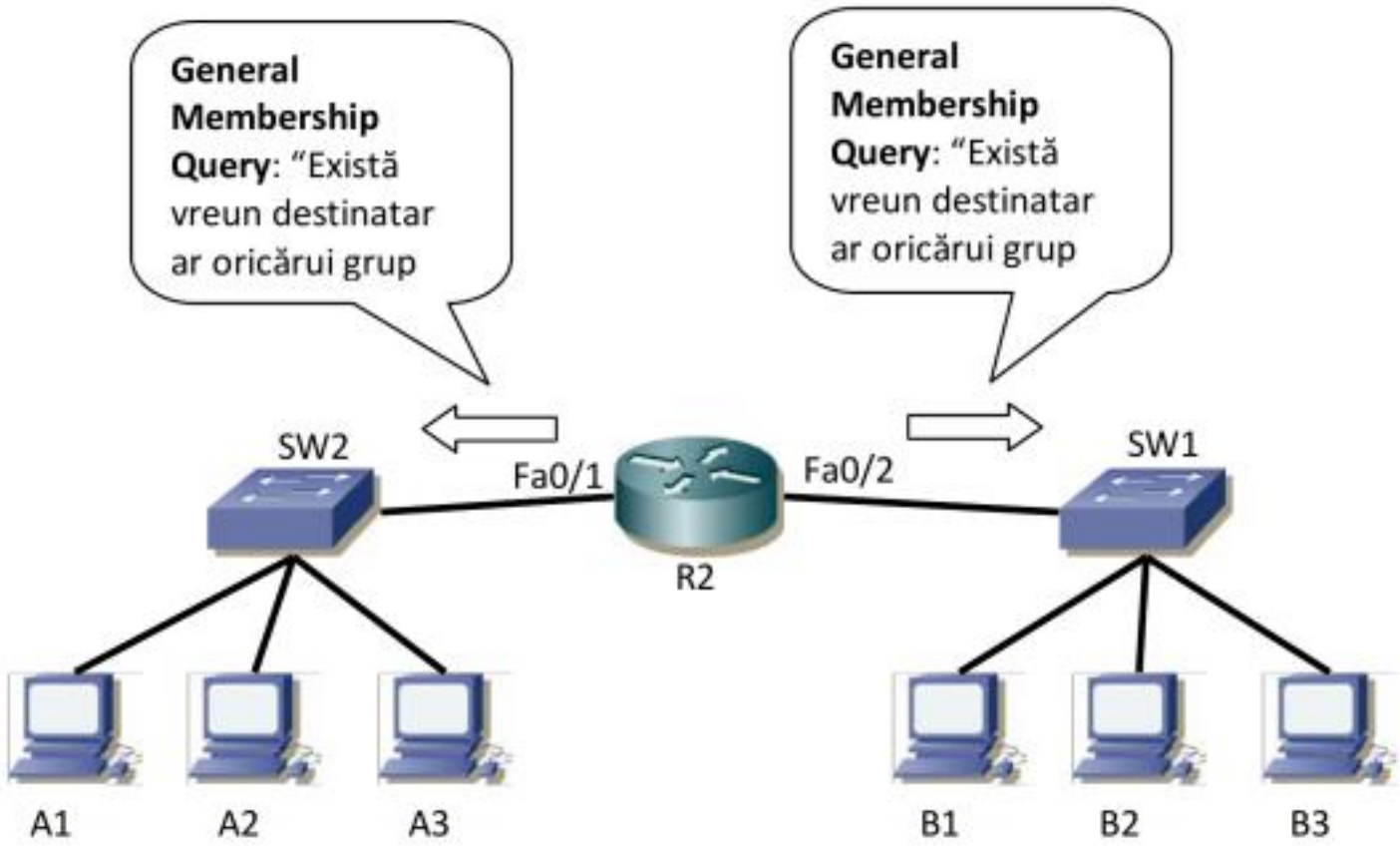

Figura 9-5: Înregistrarea la un grup multicast

1. Înainte de a se trimite orice mesaj IGMPv2, aplicația multicast client se va porni pe stațiile A1, A2 și B2 care sunt destinatarii pentru *stream*-ul de multicast trimis din Internet. Odată pornită, administratorul va trebui să configureze adresa IP multicast pe care va fi trimisa transmisia (aceeași adresă care este setată și pe componenta server a aplicației). Se va presupune adresa 227.1.1.1. În acest moment placa de rețea va asculta pe 3 adrese MAC:
   - **01.00.5E.01.01.01**, adresa MAC obținută din derivarea adresei IP 227.1.1.1 (prin procesul descris în capitolul 1.3.2);
   - **01.00.5E.00.00.01**, adresa MAC obținută din derivarea adresei IP 224.0.0.1 pe care toate echipamentele (stații și rutere) ascultă în mod implicit;
   - **BIA** (Burned In Address), adresa nativă a plăcii de rețea
2. Odată pornit și la fiecare 60 de secunde, ruterul R2 va trimite un *General Membership Query* cu adresa IP destinație: 224.0.0.1 (*all-hosts*). În acest mesaj câmpul *Group Address* va fi setat la valoarea 0.0.0.0. Cu alte cuvinte, ruterul va trimite un astfel de mesaj pe fiecare segment Ethernet la care este conectat, cu intenția de a verifica dacă există cel puțin un destinatar pentru orice grup de multicast. Cași mesajele de *Report* și *Leave*, un *General Membership Query* va avea valoarea TTL egală cu 1.

Membership Report

Un mesaj IGMPv2 Report este folosit de către o stație pentru a semnala ruterului faptul că dorește să primească trafic multicast pentru un anumit grup. Acest tip de mesaj poate fi trimis sub două forme:

- *Membership Report* **solicitat** – este trimis ca răspuns la un *Membership Query* conținând în câmpul *Group Address*, adresa grupului pentru care stația dorește să primească trafic multicast;
- *Membership Report* **nesolicitat**– este trimis imediat ce aplicația multicast a fost configurată de administrator să primească trafic pentru un grup identificat printr-o adresă IP. Acest comportament este dat de aplicație; există aplicații care așteaptă un *Membership Query* de la ruter și altele care trimit direct acest tip de mesaj pentru a face înscrierea mai rapidă.

În continuare se va trata cazul unui *report* solicitat. Un raport nesolicitat funcționează în același mod doar că nu este un răspuns la un pachet de `Query`.

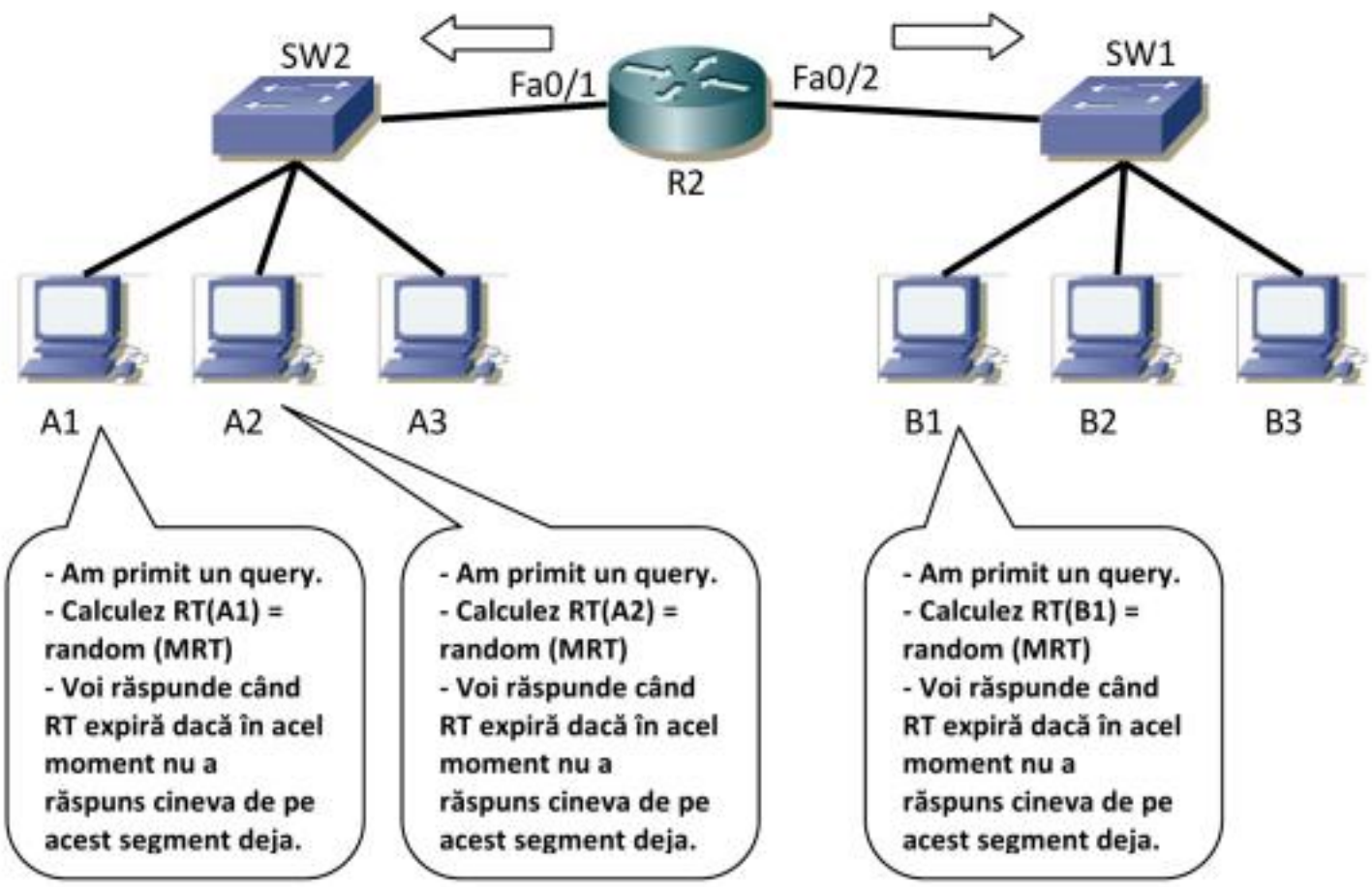

Figura 9-6: Calcularea MRT

Mesajul de *General Membership Query* va ajunge la toate stațiile din rețea căci a fost trimis cu adresa IP destinație 224.0.0.1. Toate stațiile care fac parte dintr -un grup oarecare vor dori să răspundă deoarece câmpul *Group Address* a fost setat la valoarea 0.0.0.0 (*any group*). O problemă apare în momentul în care în rețeaua locală sunt foarte multe stații care doresc un *stream* multicast. Dacă toate ar răspunde în același timp cu un *Membership Report solicitat* la fiecare 60 de secunde, ruterul ar avea foarte multe pachete de procesat. O a doua problemă este faptul că ruterul nici nu are nevoie, de fapt, să primească 40 de *Report*-uri de la 40 de stații care se află pe același segment, deoarece un singur mesaj ar fi suficient pentru a putea trimite *stream*-ul pe acea interfață Ethernet. Faptul ca sunt 40 de stații pe segment ce doresc stream-ul sau una singură nu reprezintă nicio diferență pentru ruter. Pentru a rezolva aceste două probleme se folosește câmpul MRT din *General Membership Query* alături de un comportament numit **Report Supression.**

Când un destinatar multicast primește un mesaj de *query*, el extrage valoarea exprimată în zeci de secunde din câmpul MRT [1.4.2]. În continuare, calculează o valoare numită *Response Time* după formula: RT = *random*(MRT). Protocolul va porni acum un *timer* cu valoarea RT iar când acest *timer* va expira, stația va trimite mesajul de *Membership Report*. Acest algoritm împiedică problema de rafală de mesaje Report descrisă mai sus și asigură o distribuție relativ juniformă de mesaje de Report în timp, dacă există mai multe grupuri pe același segment Ethernet.

În continuare persistă problema mesajelor redundante trimise de toate stațiile aflate pe un segment care fac parte dintr-un grup. Pentru a rezolva acest lucru, IGMPv2 specifică adresa IP destinație a unui mesaj de *Membership Report* ca fiind adresa grupului de multicast pentru care dorește să primească trafic; în cazul de față: 227.1.1.1.

Se va presupune că într-o rețea există 30 de stații ce doresc trafic pentru grupul 227.1.1.1. Când vor primi un *General Query*, toate vor calcula un RT. Stația care a obținut cel mai mic RT, va trimite un *Report* cu câmpul *Group Address* setat la valoarea 227.1.1.1 și cu adresa IP destinație setată tot la valoarea 227.1.1.1. Acest *Report* va fi primit de ruter și de toate celelalte 29 de stații, mesajul fiind trimis pe adresa grupului. Când cele 29 de stații vor primi mesajul de *Membership Report* nu vor mai trimite la rândul lor mesaje de *Report* considerând ca un singur mesaj a fost suficient pentru a ajunge la ruter. Acest comportament poartă numele de *Report Supression*.

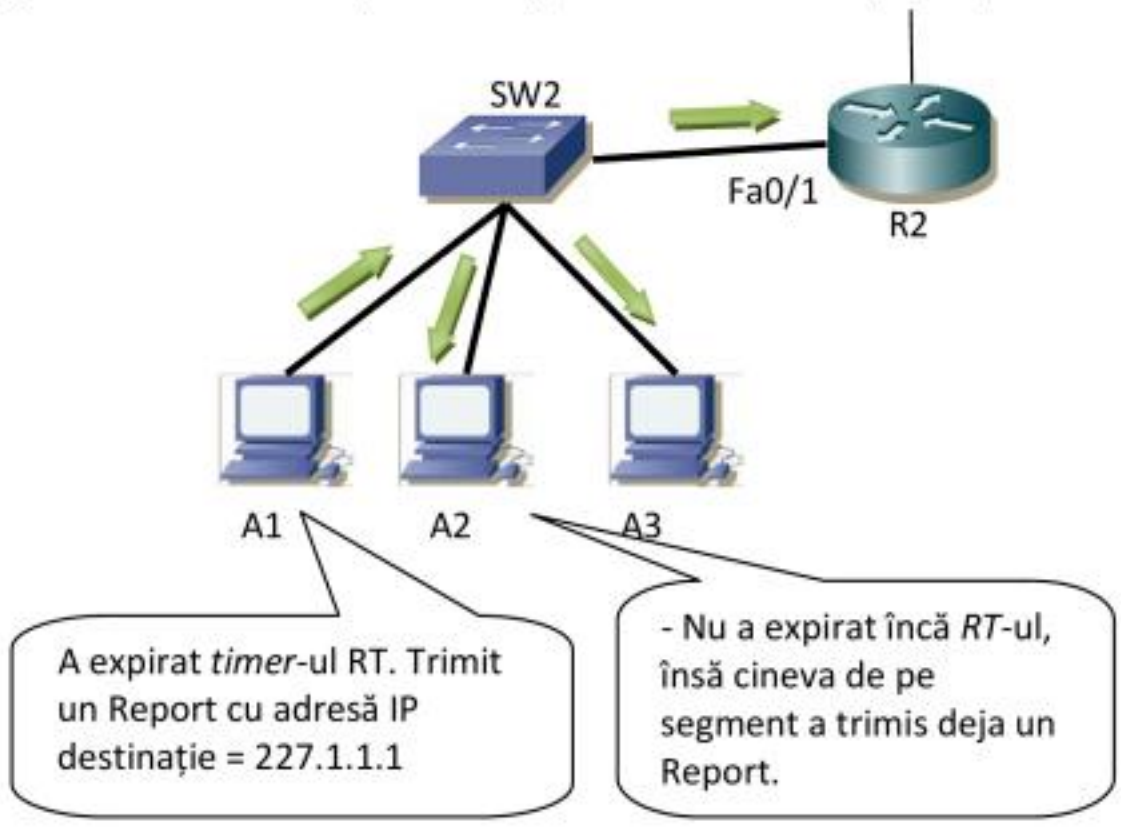

Figura 9-7: Report Supression

9.4.4 Părăsirea unui grup de multicast

Mecanismul IGMP *Leave* este unul dintre principalele motivele pentru care versiunea a doua a protocolului a fost creată. În IGMPv1, părăsirea unui grup de multicast era făcută dacă un Report nu era primit în urma trimiterii a 3 mesaje de General Membership Query. Un mesaj de query era trimis o data la 30 de secunde deci unui ruter necesita 90 de secunde pentru a concluziona că nimeni de pe segment nu mai dorește stream-ul multicast.

Ca răspuns la această metodă ineficientă, IGMPv2 include un mesaj special de părăsire a unui grup, numit *Leave Group*, prin care o stație poate informa ruterul că nu mai dorește trafic multicast. Cu acest mecanism, un ruter poate determina dacă mai există cineva pe segment care dorește *stream*-ul în doar 3 secunde.

Pentru a reveni această performanță, IGMPv2 folosește, pe lângă mesajul *Leave* și mesajul *Group-Specific Query*, prin care interoghează un segment de rețea pentru a vedea dacă mai există cel puțin un destinatar pentru un grup specific. În continuare se vor prezenta pașii din procesul de părăsire a unui grup multicast.

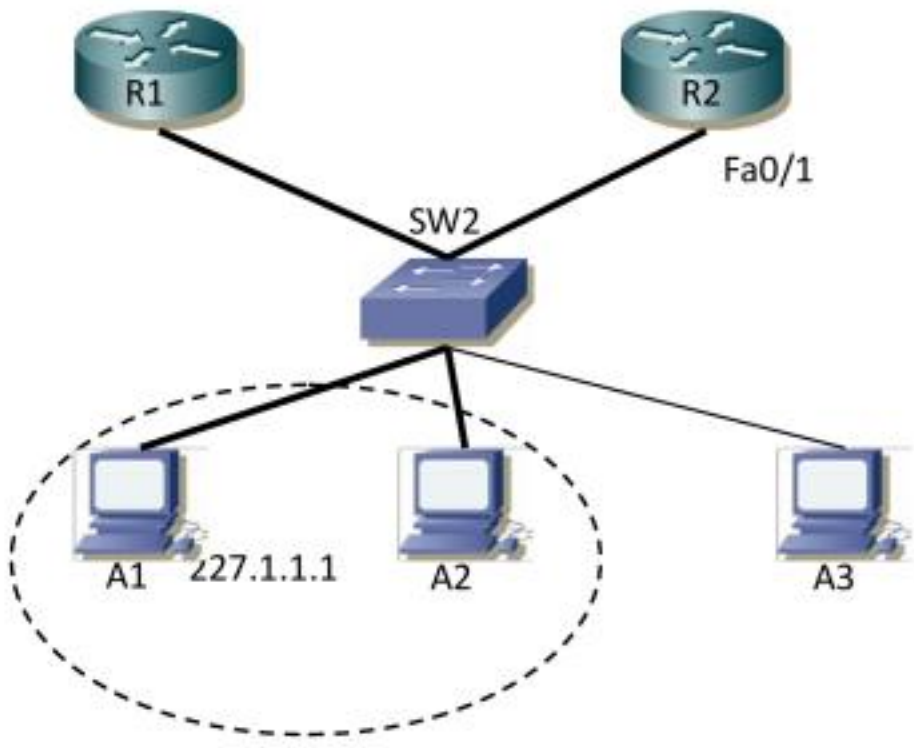

Figura 9-8: Părăsirea unui grup de multicast

1. A1 și A2 fac parte din grupul 227.1.1.1. Stația A1 dorește să părăsească grupul multicast.
2. A1 trimite un mesaj de tip IGMPv2 *Leave* către adresa IP destinație 224.0.0.2 (*all routers*) pentru a semnala tuturor ruterelor multicast de pe segment că dorește să părăsească grupul pe care l-a completat în câmpul *Group Address*.
3. Ruterul R1, responsabil cu managementul IGMP pentru grupul 227.1.1.1, va trimite un mesaj de tip *Group-Specific Query* cu câmpul *Group Address* setat la valoarea 227.1.1.1, pentru a analiza dacă mai există cineva pe acel segment care dorește să primească trafic pentru acel grup.
4. Deoarece A2 încă dorește trafic pentru acest grup, va răspunde cu un *Membership Report*. Mecanismul de *Report Supression* descris anterior se va folosi și în acest caz.
5. R1 va primi *Report*-ul și va concluziona că trebuie să trimită în continuare trafic pe acel segment.

Dacă pe segment nu ar fi existat altă stație care să necesite trafic pentru grupul 227.1.1.1, ruterul ar fi aplicat un algoritm care în maxim 3 secunde poate să concluzioneze dacă mai trebuie să trimită sau nu trafic multicast pe acel segment Ethernet. Acest algoritm definește două constante:

- *Last Member Query Interval* este o valoare de timp și este egală cu valoarea câmpului MRT. Într-un mesaj de tip *Group-Specific Query* câmpul MRT are în mod implicit o valoare de o secundă.
- *Last Member Query Count* este un număr întreg pozitiv folosit în algoritm, al cărui pseudocod e descris mai jos. Valoarea sa implicită este 2.

Când ruterul va primi un mesaj de *Leave* va trimite automat un *Group-Specific Query*. Dacă nu va primi un Report în intervalul definit de *Last Member Query Interval,* va continua să trimită mesaje *Group-Specific Query* și să aștepte intervalul *Last Member Query Interval* pentru fiecare query de numărul de ori definit de variabila *Last Member Query Count.* Pseudocodul pentru această operație este prezentat mai jos.

```
Do {
   Send a Group-Specific Query ;
   Last_Member_Query_Count -- ;
While (Last Member Query Count != 0 &&
        no_answer_in Last Member Query Interval )
```

Se poate observa ușor din cele de mai sus că bucla programatică se va repeta de 3 ori numărând și primul mesaj *Group-Specific Query* trimis ca răspuns la mesajul de *Leave*.

9.5 Optimizări ale transmiterii multicast în LAN

Se va considera următoarea topologie în care stațiile A1 și A2 au anunțat printr-un *Membership Report* faptul că vor să primească *stream*-ul 227.1.1.1.

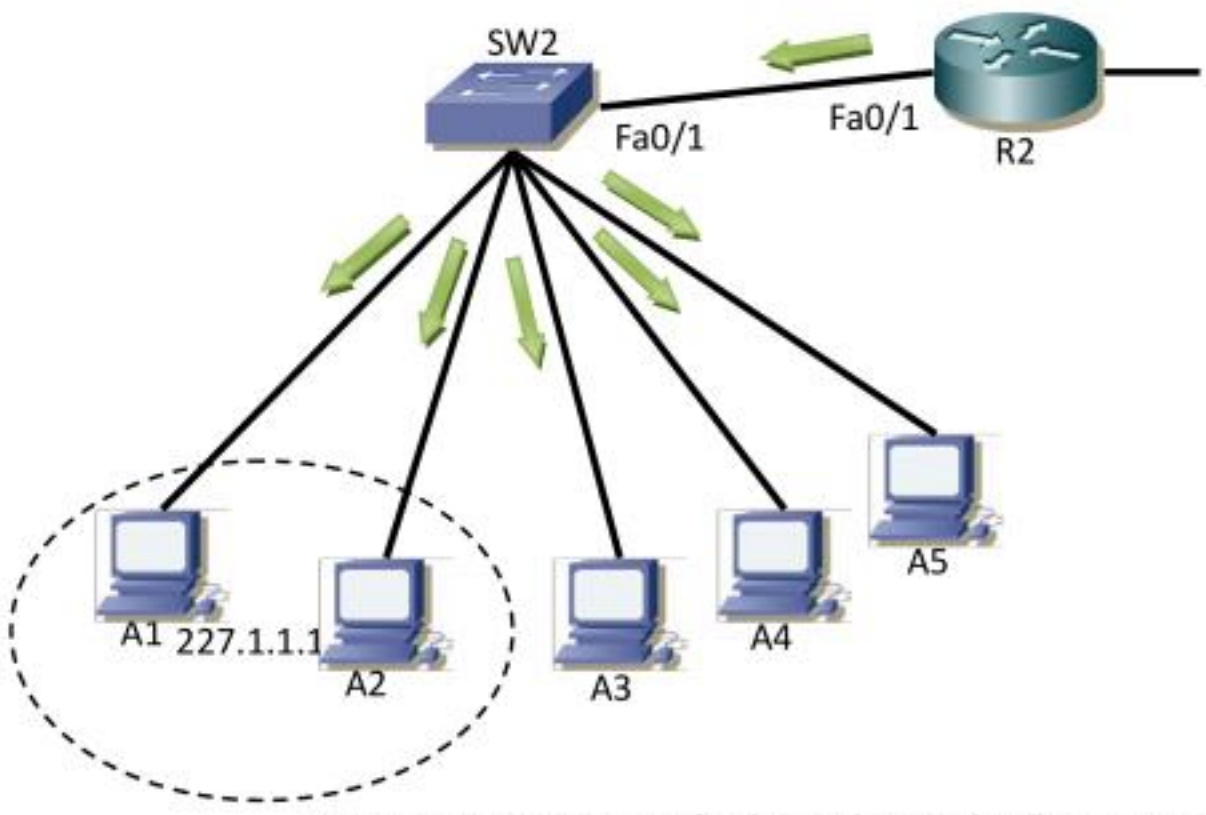

Figura 9-9: Tratarea traficului multicast de către un switch

Pentru că există cel puțin o stație pe interfața Fa0/1 a ruterul R2 care dorește să primească trafic multicast, ruterul R2 direcționează în rețeaua locală *stream*-ul primit din Internet. Când acest trafic ajunge la switchul SW2, acesta introduce cadrele multicast în procesul de comutare folosind tabela CAM. Această tabelă conține o mapare între adresele MAC sursă ale stațiilor din rețeaua locală și porturile pe care aceste stații se află. Astfel, când un cadru intră în procesul de comutare, switchul inspectează tabela sa CAM și încearcă să caute o mapare pentru adresa MAC destinație a

cadrului. Dacă această adresă nu este găsită în tabelă, switchul trimite cadrul pe toate porturile în afară de cel pe care acesta a sosit, pentru a garanta transmisia până la destinație.

Revenind la scenariul din topologie, traficul multicast ajunge la switch cu adresa MAC destinație 01.00.5E.01.01.01. Problema switchului este că acesta nu va găsi niciodată o mapare în tabela sa CAM pentru această adresă sau orice altă adresă multicast, pentru că acestea nu se asignează niciodată ca adrese sursă. Concluzia este ca switchul va trata orice trafic multicast în același mod: îl va trimite pe toate interfețele sale. În cazul în care este vorba de o rețea locală de 100 de stații, însă doar 2 din ele fac parte din grupul multicast, ocuparea inutilă a rețelei nu este de dorit.

Pentru a rezolva această problemă se folosesc două protocoale: *Cisco Group Management Protocol* (CGMP) și *IGMP Snooping*. Prezentarea în detaliu a celor două protocoale depășește scopul acestei cărți astfel că în cele ce urmează se va face o trecere sumară prin funcționarea de bază a fiecărui protocol.

9.5.1 Cisco Group Management Protocol

CGMP este un protocol proprietar Cisco ce funcționează la nivelul 2. Pentru a-și îndeplini scopul acesta trebuie să ruleze atât pe ruter cât și pe switch. Ideea după care CGMP a fost creat presupune o comunicare între ruter și switch prin care ruterul îi indică switchului adresele MAC ale stațiilor care vor să primească multicast. Ruterul cunoaște aceste adrese MAC din comunicația IGMP pe care o are cu fiecare stație ce dorește să facă parte din grupul de multicast.

Odată cu adresele MAC ale stațiilor dintr-un anumit grup, switchul primește de la ruter și adresa MAC multicast a grupului. Astfel acesta caută în tabela CAM porturile pe care cunoaște adresele MAC ale stațiilor și creează mapări noi între adresa MAC de multicast a grupului și porturile fiecărei stații.

9.5.2 IGMP Snooping

IGMP Snooping este o alternativa la stilul Cisco pentru problema *flooding*-ului[1] pe care switchul îl face în mod implicit cu orice trafic multicast. Cea mai mare diferență între acest protocol şi CGMP este că IGMP Snooping nu trebuie rulat decât pe switch. Acest comportament i-a câştigat în timp popularitate față de CGMP, care ocupă mai multă bandă şi implică procesare din partea ruterului.

IGMP Snooping îi oferă switchului „inteligența" de care are nevoie pentru a putea interpreta mesajele IGMP pe care le comută între stații şi ruter. Astfel switchul poate să învețe pe ce porturi se află stațiile ce fac parte dintr-un anumit grup şi să evite procesul de *flooding*. Pe lângă această funcționalitate de bază, sunt adăugate şi diferite optimizări pentru IGMP. Spre exemplu când switchul primeşte un mesaj IGMP *Leave* pentru un anumit grup, el va căuta în tabela CAM și va decide dacă mai există cel puțin o stație în grupul respectiv. În caz afirmativ nu va mai comuta mesajul de *Leave* la ruter pentru a evita schimbul de mesaje *Group-Specific Query* şi *Membership Report* pe care ruterul şi stațiile ar urma să îl fac în urma unui mesaj IGMP *Leave*.

În present, IGMP Snooping este protocolul preferat atât de Cisco cât şi de alți producători pentru că resursele de rețea consumate sunt mult mai mici şi pentru că adaugă mai multe optimizări pentru IGMP.

---

[1] http://www.ciscopress.com/articles/article.asp?p=101629&seqNum=2

9.6 Rutarea multicast

Principalul atribut al unui protocol de rutare **unicast** este propagarea rutelor în mod dinamic între mai multe rutere. Pentru a atinge acest scop, orice IGP (*Interior Gateway Protocol*) porneşte prin a anunța rețelele sale direct conectate tuturor vecinilor săi pentru ca aceştia să le anunțe mai departe. Astfel după un anumit timp de convergență ce depinde de protocolul folosit, toate ruterele vor cunoaşte cea mai scurtă cale către orice destinație, pe baza metricii folosite.

Din acest comportament reiese faptul că un protocol de rutare IGP unicast nu ar putea niciodată să direcționeze trafic multicast din cauza faptului că adresele de multicast nu se asignează niciodată pe interfețe.

Un protocol de rutare multicast nu este interesat de propagarea rețelelor direct conectate, ci de construirea celei mai bune căi de la sursa de multicast la destinația de multicast peste o rețea de nivel 3. Există două tipuri de protocoale de rutare multicast folosite în implementări reale:

- Protocoale *dense* – sunt preferate în momentul în care există cel puțin un destinatar în fiecare subrețea dintr-o anumită arie geografică.
- Protocoale *sparse* – sunt preferate în momentul în care destinatarii multicast sunt grupați în puține subrețele dintr-o anumită arie geografică.

În prezent cele mai cunoscute protocoale de rutare multicast sunt: *Protocol Independent Multicast Dense Mode* (PIM-DM), *Protocol Independent Multicast Sparse Mode* (PIM-SM), *Multicast OSPF* (MOSPF), *Distance Vector Multicast Routing Protocol* (DVMRP). Dintre acestea PIM-SM şi PIM-DM conduc detaşat în topul de popularitate, MOSPF şi DVMRP nefiind nici măcar suportate complet pe rutere Cisco.

9.6.1 PIM-DM

Motivul pentru care protocolul PIM (Protocol Independent Multicast) este atât de popular în ambele forme ale sale (Dense Mode și Sparse Mode) este pentru că este se bazează pe implementarea protocoalelor de rutare unicast pentru a găsi cea mai scurtă cale către destinație/sursă. După cum s-a menționat în subcapitolul anterior [1.6], construirea unei căi optime între destinație și sursă este unul dintre scopurile principale ale unui protocol de rutare multicast. PIM evită „reinventarea roții" și folosește orice rută din tabela de rutare pentru a determina calea optimă fără a lua în considerare ce protocol de rutare unicast a generat acea rută. Dezavantajul competitorului DVMRP, în acest caz, este că aduce *overhead*-ul unei implementări interne de protocol de rutare unicast, foarte asemănător cu RIPv2, care să facă determinarea căii. De cealaltă parte, MOSPF introduce o dependență de protocolul de rutare unicast OSPF pentru a putea funcționa. Astfel, ca și în multe alte situații din lumea rețelelor de calculatoare, simplitatea câștigă.

PIM a fost la început un protocol proprietar Cisco, însă a fost deschis către standardizare prin RFC-urile 2362, 3446 si 3973. Asemenea altor protocoale de rutare, acesta folosește mesaje *hello* trimise la adresa 224.0.0.13 și un proces de creare de adiacențe. Aceste detalii depășesc însă scopul acestei cărți și nu vor fi aprofundate

Protocoalele *dense* presupun în funcționarea lor că aplicația multicast folosită este atât de populară încât există cel puțin un destinatar în fiecare subnet al ariei de transmisie. Din acest motiv un ruter care rulează un protocol *dense* va trimite în mod implicit traficul multicast pe toate interfețele în afară de interfața pe care acesta a venit. Dacă funcționarea *dense* ar fi însă limitată la acest comportament, traficul multicast ar traversa tot Internetul. Din această cauză, în orice protocol *dense* există posibilitatea ca un ruter să poată refuza traficul multicast pentru un anumit

grup pentru a evita extinderea *stream*-ului în direcții în care acesta nu are destinatari. Înainte de detalierea acestei operații se vor defini:

- *router upstream* – este un ruter vecin care se află mai aproape de sursă decât ruterul în discuție;
- *router downstream* – este un ruter vecin care se află mai departe de sursă decât ruterul în discuție.

Există două situații în care un ruter multicast nu va dori traficul multicast de la un ruter *upstream*.

- Ruterul în discuție nu are niciun vecin *downstream* care să dorească traficul multicast;
- Ruterul nu cunoaște nici o stație dintr-o subrețea direct conectată care să dorească traficul multicast.

Dacă ambele condții de mai sus se aplică, ruterul va trimite un mesaj numit *prune* ruterului *upstream,* informându-l să nu mai trimită trafic multicast către el. Astfel se realizează o restrângere a ariei în care *stream*-ul e transmis, pentru a trimite doar către destinațiile valide.

Comportamentul descris mai sus este comun tuturor protocoalelor *dense* și este respectat și de PIM-DIM.

9.6.2 Reverse Path Forwarding Check

Chiar dacă există un mecanism în protocoale *dense* și implicit în PIM-DIM, prin care un ruter poate specifica unui vecin *upstream* faptul că nu dorește trafic multicast, da că într-un graf un nod trimite în mod implicit trafic pe toate muchiile în afară de cea pe care traficul a venit, vor apărea în mod inevitabile bucle. Pentru a nu avea această problemă, înainte de a trimite trafic în toate direcțiile, un ruter analizează adresa IP sursă a pachetelor multicast și caută rețeaua corespunzătoarea acelei adrese în tabela de rutare. Dacă în tabela de rutare, ruterul are ca interfață de ieșire către adresa de rețea determinată aceeași interfață pe care a primit pachetele multicast, stream-ul este unul valid și va fi trimis pe toate interfețele în afară de cea pe care a sosit. Dacă însă ruta către sursa traficului este pe o interfă diferită de interfața pe care pachetele multicast au venit la ruter, acesta le aruncă și trimite un mesaj *prune* vecinului pentru a îi spune să nu mai trimită trafic pe acea legătură căci îl trimite în buclă.

Figura 1-8 prezintă o topologie în care mesajul *prune* este folosit pentru a limita transmiterea traficului multicast în PIM-DM.

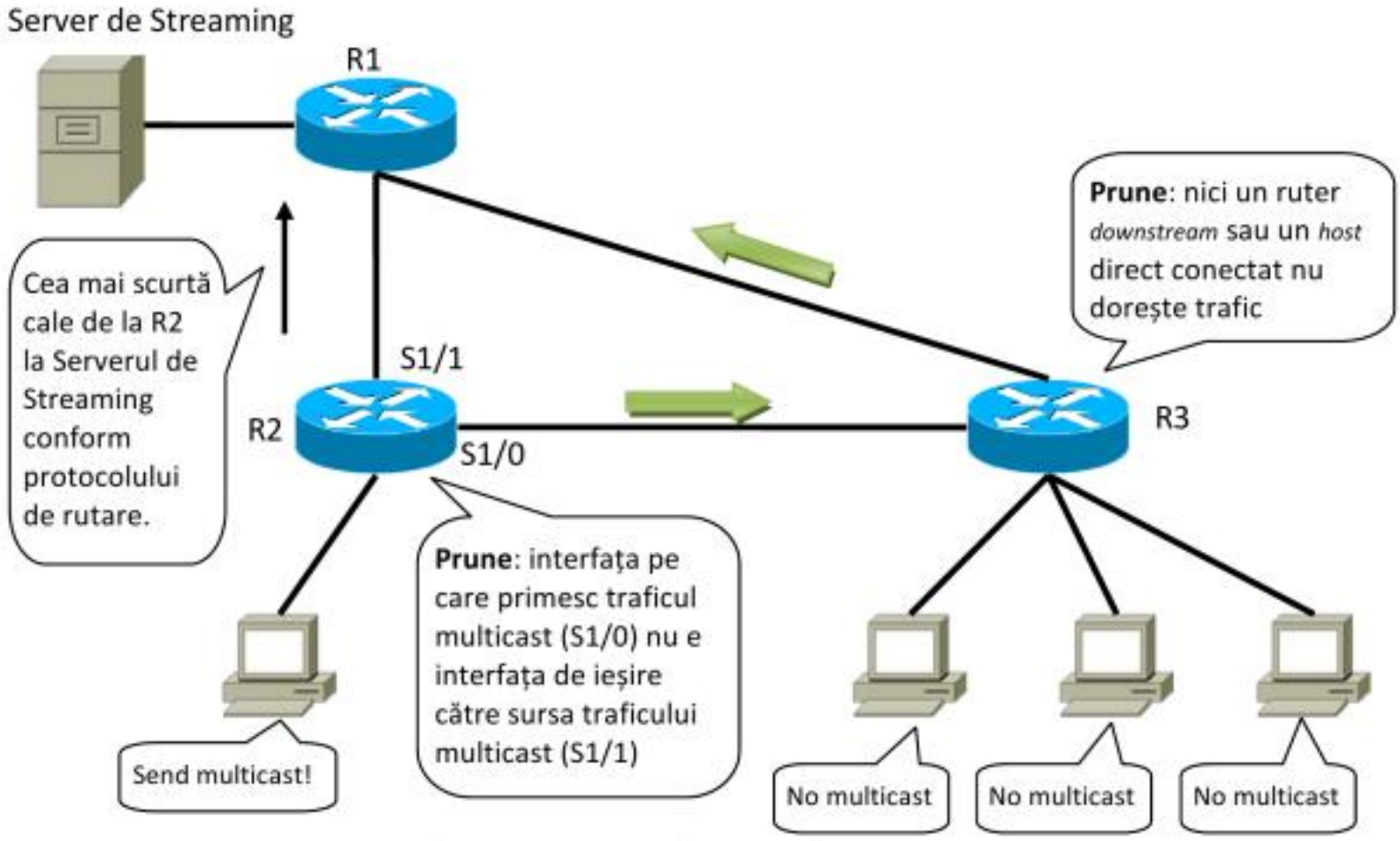

Figura 9-10: Mesaje prune în PIM-DM

În topologia de mai sus ruterele R2 și R3 trimit mesaje de *prune*, însă din motive diferite. În pașii ce urmează se vor analiza aceste motive pentru fiecare dintre cele două rutere:

1. Ruterul R1 primește *stream*-ul de la server și pentru că rulează PIM -DM, trimite traficul pe toate interfețele în afară de cea pe care l-a primit.
2. Ruterul R3 primește *stream*-ul de la R1 și aplică testul RPF. Pentru că cea mai scurtă cale a lui R3 către server este prin R1, testul este trecut cu succes. Prin urmare R3 aplică comportamentul *dense* și trimite pachetele multicast spre ruterul R2, considerat deocamdată de către R3, un ruter *downstream*.
3. Ruterul R2 primește *stream*-ul de la ruterul R3 și aplică testul RPF. Pentru că cea mai scurtă cale de la R2 la sursă este prin ruterul R1, testul este picat. Astfel R2 trimite un mesaj de *prune* ruterul R3.
4. Ruterul R3 nu este direct conectat la nicio stație care să dorească *stream*-ul iar R2 tocmai a specificat faptul că nu dorește trafic multicast de la el. Neavând alt destinatar pentru trafic, R3 trimite un mesaj de *prune* către R1 spunându-i acestuia să nu mai trimită *stream*-ul.
5. Ca și R3, și R2 primește pachetele multicast de la R1, însă pentru că este direct conectat la o stație care dorește acele pachete și pentru că testul RPF către R1 este trecut cu succes, acceptă traficul.

Când un mesaj de tip *prune* este primit de către un ruter ce rulează PIM-DM, acesta își pune interfața în modul *pruned* (nu mai trimite trafic multicast)și o păstrează astfel timp de 3 minute. După acest interval ruterul scoate interfața din modul acest mod și începe să trimită din nou multicast pe ea. Dacă ruterul *downstream* în continuare nu are nicio destinație căreia să îi trimită multicast, acesta va trebui să trimită din nou un mesaj *prune* și să continue acest proces din 3 în 3 minute.

9.6.3 Source-Based Trees

Comportamentul implicit pe care PIM-DM îl are în legătură cu trimiterea traficului în toate direcțiile și a îl restricționa mai apoi cu mesaje *prune* creează ceea ce standardul numește un

source-based tree sau *shortest-path tree* (SPT). Acest arbore definește cea mai scurtă cale de la sursă către toate destinațiile, având sursa ca rădăcină. Pentru fiecare grup de multicast și fiecare sursă diferită, va exista un SPT definit astfel (IP_SURSĂ, IP_GRUP). Aceasta este notația folosită atât în standard cât și în *output*-ul comenzilor Cisco IOS. Acești arbori sunt cu adevărat relevanți în comparația dintre *dense* și *sparse* și vor fi detaliați într-un subcapitol ulterior.

9.6.4 Mesajul *graft*

Situația prezentată în topologia de mai jos este o continuare a procesului din figura 1-8. Ruterul R3 trimise un mesaj de tip *prune* iar R1 își setase interfața S1/1 în modul *pruned*.

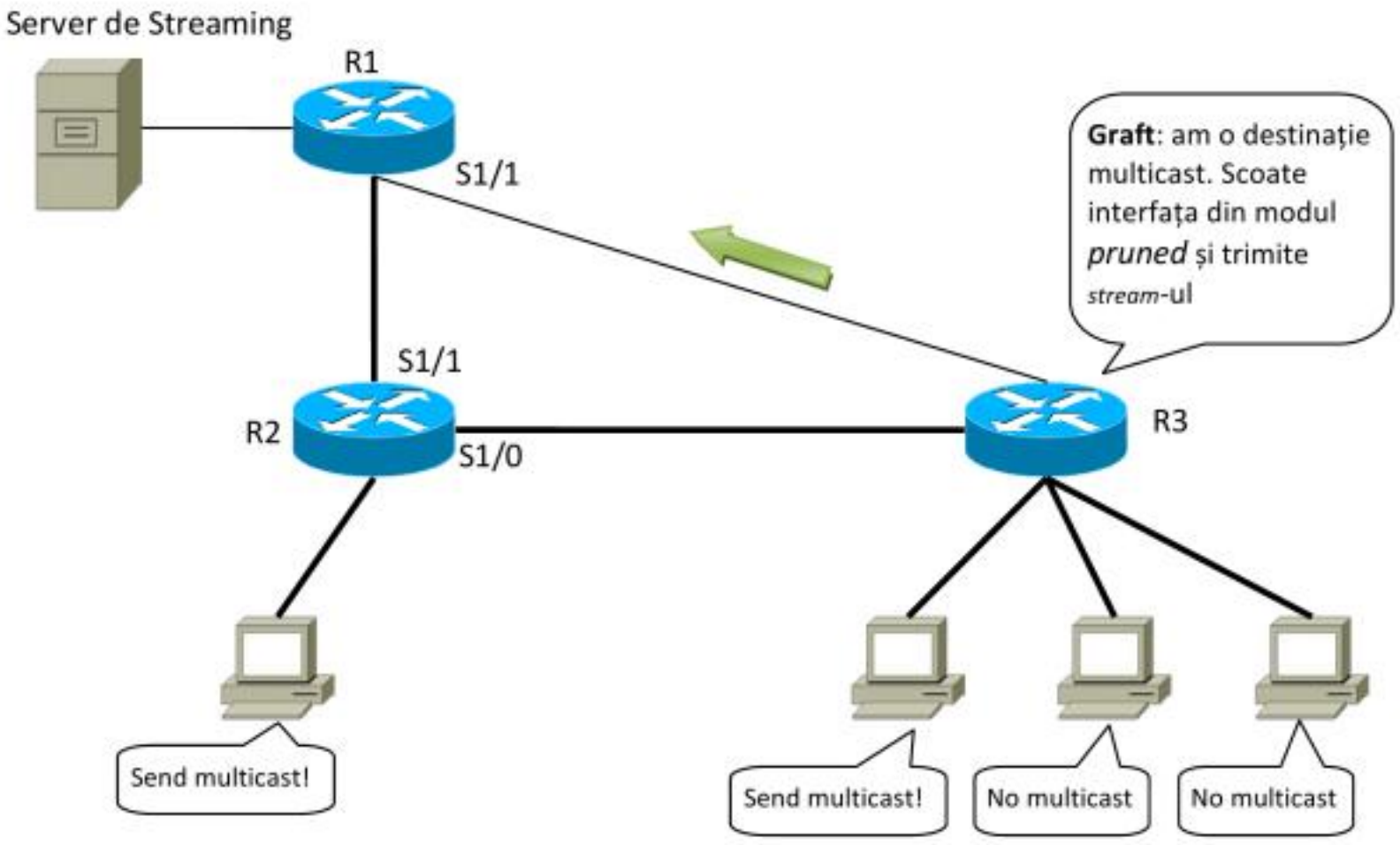

Figura 9-11: Mesajul graft în PIM-DM

Una dintre stațiile direct conectate la ruterul R3 trimite un mesaj IGMP *Membership Report* pentru primi trafic pentru grupul în care serverul de *streaming* trimite multicast. Ruterul R3 concluzionează astfel că există un destinatar pentru traficul de multicast pentru care el a trimis un mesaj de *prune*. Pentru ca stația din LAN să nu aștepte un timp maxim de 3 minute înainte de putea primi *stream*-ul, ruterul R3 trimite un mesaj de tip *graft* lui R1. Când ruterul R1 primește acest mesaj, el scoate automat interfața sa din modul *pruned* și începe să trimită trafic cât timp *stream*-ul este încă activ sau până când va primi iarăși un mesaj *prune* (ex: cazul în care stația ar trimite un IGMP *Leave-Group*).

9.6.5 PIM-SM

Protocoalele *sparse* sunt recomandate în standard spre a fi folosite în momentul în care destinațiile multicast sunt localizate în puține subrețele din aria geografică. PIM *Sparse Mode* a fost optimizat însă de Cisco și este în momentul de față singurul protocol multicast pe care Cisco îl recomandă pentru *orice fel* de topologie. Metodologia pe care *sparse* o folosește în distribuția multicast este complet opusă celei *dense*. Dacă un ruter ce rulează PIM-SM primește trafic multicast pentru un anumit grup, nu îl va trimite pe niciun port până când un ruter *downstream* sau o stație dintr-o rețea direct conectată nu va cere acest trafic. Dacă în PIM-DM era nevoie de un mesaj *prune*

pentru a opri trimiterea traficului, în PIM-SM este nevoie de un mesaj *join* pentru a începe trimiterea acestuia.

Starea implicită a unei interfețe de ruter multicast *sparse* este *not forwarding*. Odată ce se primește un mesaj *join* pe o interfață, aceasta în modul *forwarding* timp de 3 minute și trimite trafic multicast pe acea legătură. După cele 3 minute *join*-ul va trebui realizat din nou. În protocolul real există optimizări pentru a nu avea un *stream* intermitent o dată la 3 minute.

Transmiterea de trafic multicast în modul *sparse* are 2 pași:

- Transmiterea de trafic de la ruterul direct conectat la sursa multicast la un ruter special numit **Rendezvous Point** (RP);
- Transmiterea de trafic de la RP la destinațiile multicast din diferite subrețele.

RP-ul este cel mai important concept din transmisia multicast *sparse*. Traficul multicast transmis de la sursă va fi trimis mai întâi către RP, iar apoi RP-ul va direcționa pachetele către diferitele destinații multicast. O topologie multicast *sparse* poate fi văzută astfel ca având două secțiuni: un SPT pentru fiecare grup/sursă care are ca rădăcină sursa multicast și ca unică frunză RP-ul și un *shared tree* sau *Root-Path Tree* (RPT) prin care RP-ul trimite trafic către toate destinațiile dintr-un grup sau mai multe. Acest RPT este simbolizat prin (*, IP_GRUP). Cu alte cuvinte, pentru destinațiile finale, traficul va veni mereu prin punctul central RP indiferent de sursa multicastului.

9.6.6 Funcționarea PIM-SM: SPT și RPT

Se va presupune următoarea topologie:

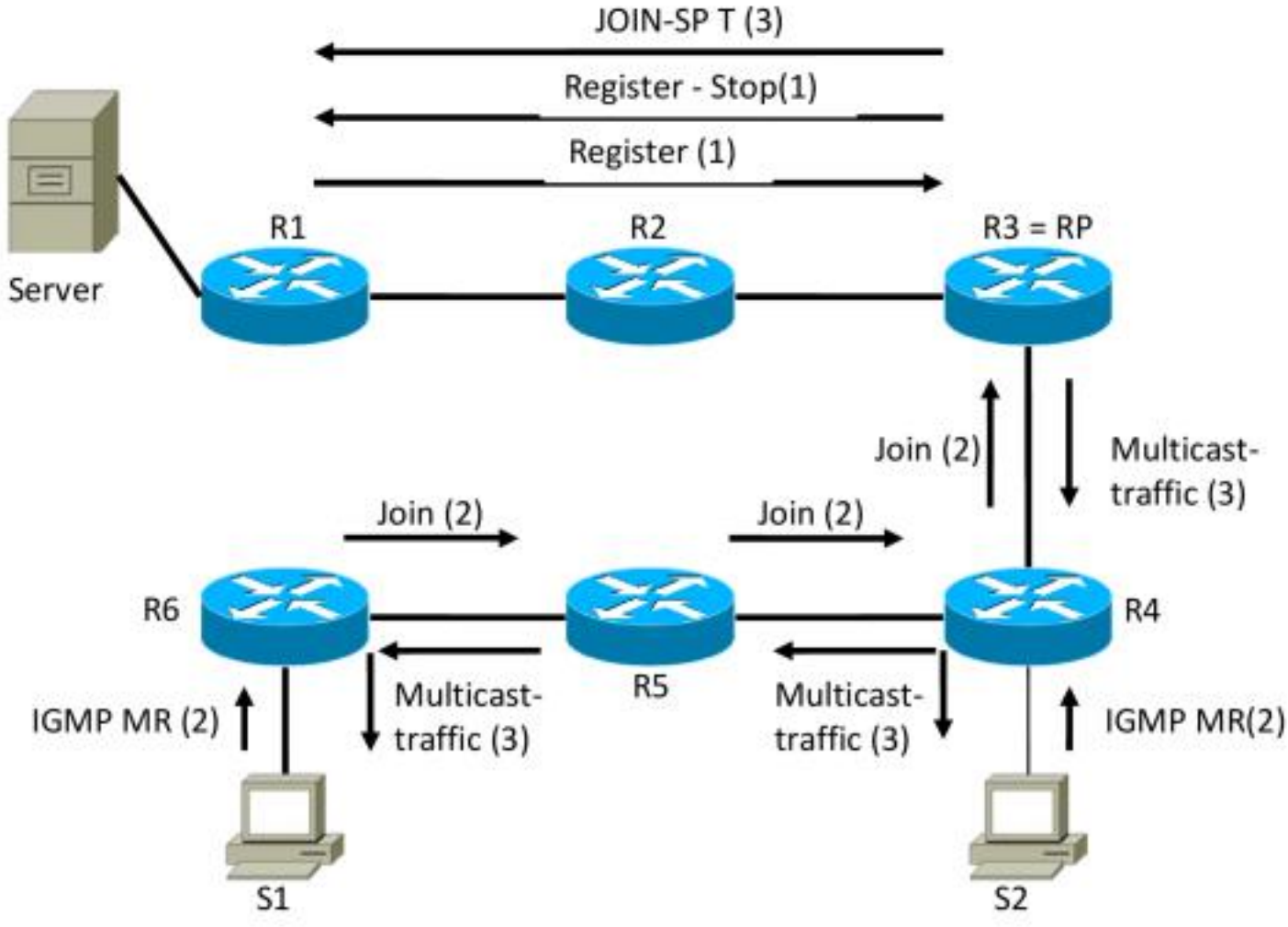

Figura 9-12: Crearea SPT-ului în PIM-SM

Înainte ca o topologie *sparse* să poată funcționa, toate ruterele din rețea trebuie să cunoască cine este RP-ul. Acest lucru se poate face static, configurând adresa RP-ului pe fiecare ruter, sau dinamic prin protocolul *Auto-RP* sau *Bootstrap Router*(BSR). În capitolul de configurări de bază se va prezenta configurarea statică a unui RP.

Pe figura de mai sus evenimentele au fost secvențializate folosind un număr de secvență ce urmează fiecărui mesaj.

Serverul direct conectat la ruterul R1 începe să trimită un *stream* pentru grupul 227.1.1.1. R1 va reacționa imediat la această acțiune și va începe să trimită mesaje **Register** către RP(R3) anunțându-l că există un stream activ la care el este direct conectat. Mesajele *Register* sunt trimise ca unicast către RP, însă în câmpul de date conțin chiar pachetele de multicast ale *stream*-ului. RP-ul va primi mesajul dar pentru că nu cunoaște nicio destinație validă pentru acest grup (nu a primit niciun *join* pentru 227.1.1.1), îl va arunca și va trimite înapoi un mesaj **Register-Stop** prin care îl va opri pe R1 din procedura de înregistrare timp de 1 minut. După acest interval R1 va începe să trimită iarăși *Register*.

Cât timp RP-ul refuză procedura de *register*, stațiile S1 și S2 trimit un mesaj de IGMP *Membership Report* pentru grupul 227.1.1.1. R6 și R4 reacționează trimițând un mesaj *join* către RP. Odată cu primirea unui mesaj de *join* pentru grupul 227.0.0.1, RP-ul nu va mai trimite *Register-Stop* către R1, iar la următorul mesaj *Register*, va extrage pachetul multicast din cel unicast și îl va trimite pe interfața către R4. În același timp va începe procedura de „înregistrare la SPT" pentru a crea un drum multicast *end-to-end*. Cu alte cuvinte RP-ul va începe să trimită mesaje *join* către R1 pentru ca acesta din urmă să poată să trimită direct *stream*-ul multicast (fără a îl încapsula în unicast). Acest ultim pas este necesar din cauza regulii de bază din modul *sparse*: un ruter nu poate trimite multicast decât dacă acesta e cerut printr-un mesaj *join*.

Astfel se creează doi arbori ce compun topologia multicast *sparse*. Arborele de la sursă la RP este un SPT pentru că este diferit funcție de locația sursei. Arborele de la RP la destinații este un RPT pentru că este partajat de mai multe grupuri care folosesc același ruter ca RP.

9.6.7 Optimizarea PIM-SM: SPT switchover

În continuarea scenariului din figura 1-12 se va adăuga o legătură între ruterele R1 și R6.

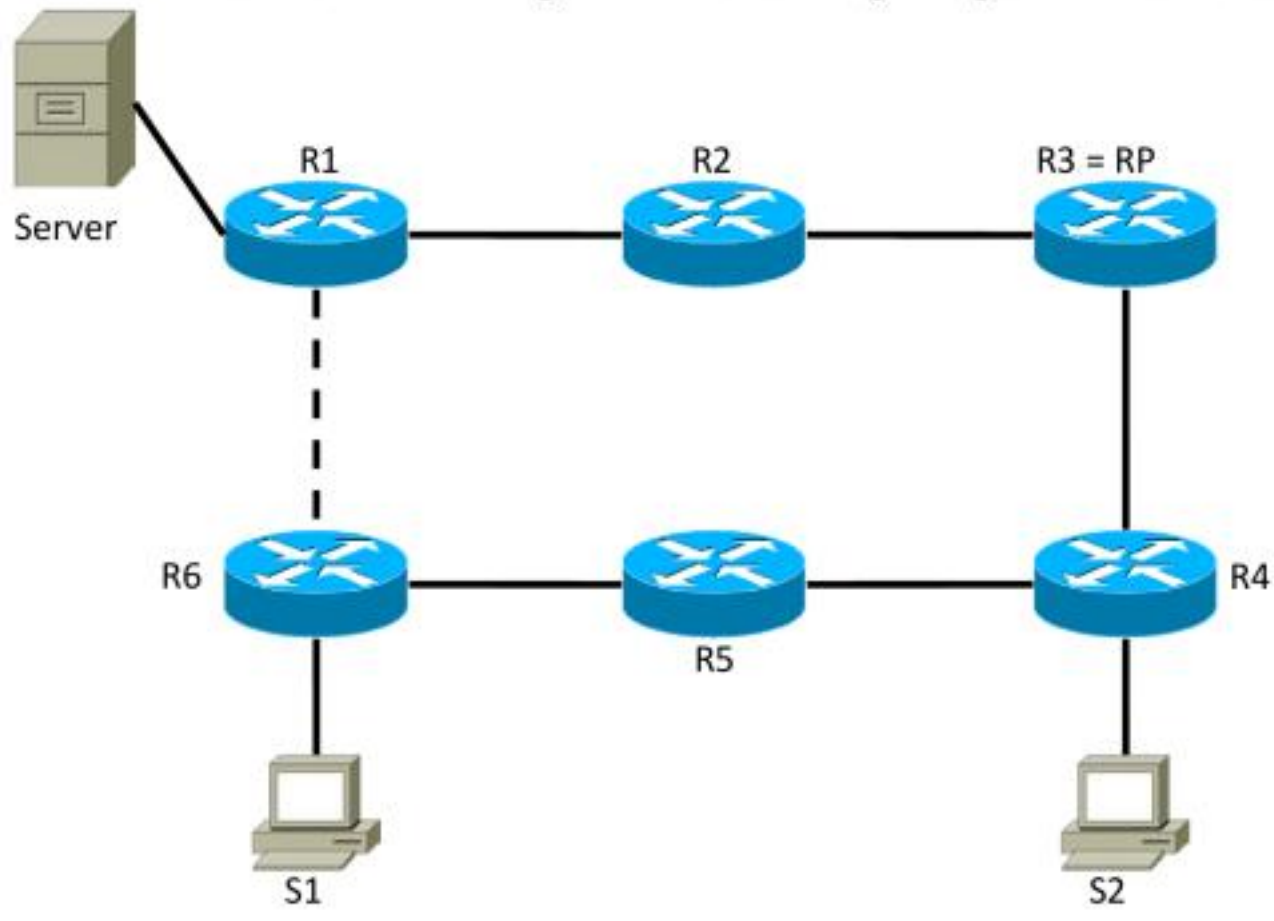

Figura 9-13: SPT Switchover

Traficul multicast este transmis acum de la R1 către RP pe un *Shortest-Path Tree*, iar apoi de la RP la stațiile S1 și S2 pe un *Root-Path Tree*. Pentru a ajunge de la R1 la R6, pachetele multicast

trebuie să treacă prin 6 hopuri. În PIM-SM, comportamentul RP-ului de a putea face *join* pe un SPT este posibil pe orice ruter. În cazul acesta, R6 se va uita în tabela de rutare și va vedea că ruta prin R1 este mult mai bună decât ruta care îl duce prin RPT cu vecinul R5. El va trimite un mesaj de *join* către R1 și se va înregistra la un SPT care să aducă traficul multicast doar peste un hop. Procedura va funcționa însă acum R6 va primi pachete multicast atât prin SPT cât și prin RPT. Pentru a folosi doar SPT-ul, R6 va trimite un mesaj *prune* către RP spunându-i că nu mai dorește pachete pentru grupul 227.1.1.1 de la sursa IP_Server. Pentru că R4 încă face *join* pe acest grup, RP va continua să trimită trafic pe interfața sa către R4, însă R5 va vedea mesajul *prune* de la R6 și va face și el *prune* astfel încât traficul prin RPT va ajunge doar la R4.

Această optimizare este de fapt motivul pentru care Cisco recomandă PIM-SM pentru orice fel de topologie. Dacă exista 100 de surse grupate în același subrețele și încă 10 ce se află în direcția diametral opusă, se va plasa RP-ul aproape de cele 100, iar celelalte stații vor face *switchover* pe SPT-ul către sursa multicast. Astfel nu mai contează dacă sursele sunt grupate sau nu dispersate/grupate, algoritmul va funcționa întotdeauna optim cât timp cât RP -ul este selectat eficient.

9.7 Configurări de bază

În protocoalele multicast partea teoretică ce presupune înțelegerea conceptelor este întotdeauna mult mai complexă decât cea de configurare. Activarea protocoalelor PIM sau IGMP se face dintr-o singură comandă, urmând ca implementarea din spatele protocolului să preia toată munca de funcționare.

Pentru a activa PIM în *dense-mode* se va folosi comanda din modul interfață:

```
(config-if)# ip pim dense-mode
```

În cazul *sparse-mode*, pe lângă activare, trebuie specificată și identitate a (adresa IP) RP-ului. Acest lucru poate fi configurat static sau dinamic prin Auto-RP sau BSR. Setarea dinamică a RP-ului depășește însă scopurile acestei cărți, un scurt exemplu fiind dat doar în capitolul de studiu de caz.

Pentru configurarea PIM *sparse-mode* în mod static se folosesc comenzile:

```
(config)# ip pim rp-address <ip_address>
(config-if)# ip pim sparse-mode
```

Un ruter Cisco are inclus în sistemul de operare posibilitatea de a se comporta ca și un client de multicast. Pentru a face acest lucru trebuie configurat IGMP la nivel de interfață:

```
(config-if)#ip igmp join-group 225.1.1.1
```

Odată cu aplicarea acestei comenzi ruterul va începe să trimită mesaje de *Membership Report* sau *Membership Leave* și va putea să facă parte din grupul de multicast 225.1.1.1.

Atenție! Pentru ca ruterul vecin să poată răspunde la cereri IGMP din partea clientului, acesta trebuie să aibă PIM activat pe respectiva interfață.

În contextul optimizării multicast în rețeaua locală, *IGMP snooping* oferă posibilitatea activării pe un VLAN specific al acestei funcționalități:

```
(config)# ip igmp snooping [vlan x]
```

Un protocol de rutare multicast determină pe ce interfață să trimită copii ale pachetelor multicast primite. Toate informațiile necesare pentru a ruta pachetele multicast sunt re ținute în

tabela multicast care mai este numităși *tabela de rutare multicast*. Aceste informații pot fi afișate folosind comanda **show ip mroute**.

Sintaxa comenzii este:

```
    show ip mroute [group-name | group-address] [source] [summary]
[count] [active kbps]
```

În continuare se vor urmări exemple de *output* ale acestei comenzi pe care se vor discuta diferiții parametrii.

```
    Router# show ip mroute 233.1.1.1
    IP Multicast Routing Table
Flags: D - Dense, S - Sparse, C - Connected, L - Local, P - Pruned
       R - RP-bit set, F - Register flag, T - SPT-bit set
Timers: Uptime/Expires

Interface state: Interface, Next-Hop, State/Mode
    (*, 233.1.1.1), uptime 0:57:31, expires 0:02:59, RP is 0.0.0.0,
    flags: DC
  Incoming interface: Null, RPF neighbor 0.0.0.0
  Outgoing interface list:
    Ethernet0, Forward/Dense, 0:57:31/0:02:52
    FastEthernet1, Forward/Dense, 0:56:55/0:01:28
    FastEthernet2, Forward/Dense, 0:56:45/0:01:22

    (172.16.16.1/32, 233.1.1.1), uptime 20:20:00, expires 0:02:55,
    flags: C
  Incoming interface: FastEthernet1, RPF neighbor 10.20.30.1
  Outgoing interface list:
    Ethernet0, Forward/Dense, 20:20:00/0:02:52
```

Pentru comanda de mai sus se vor descrie mai întâi parametrii, iar apoi interpretarea pentru tabela de rutare multicast.

9.7.1 Parametrii

Flag-uri:

- D – *Dense*: Intrarea operează în modul *dense*;
- S – *Sparse*: Intrarea operează în modul *sparse*;
- C – *Connected*: Un membru al grupului multicast este prezent pe o interfață direct conectată; Un calculator se conectează folosind mesaje IGMP.
- L – *Local*: Un ruter este membru al unui grup multicast;
- P – *Pruned*: O rută a fost trecută în starea *pruned*. Cisco IOS păstrează această informație în cazul în care un membru *downstream* dorește să se alăture sursei.
- R-*RP-bit* setat: Intrarea (S,G) indică către *Rendevous Point (RP)*. RP este de obicei o stare *pruned* in interiorul arborelui pentru o sursă.
- F – *Register* : Indică faptul că are loc o înregistrare pentru sursă multicast.
- T – *SPT-bit* setat: Pachetele au ajuns pe drumul cel mai scurt.
- H – *Hardware stwitched*: Interfața de ieșire *este Hardware Switched* datorita IP multicast MLS.

Timers: sunt de forma *Uptime/Expires*.

Interface state: indică starea interfețelor de intrare și de ieșire:

- *Interface*: tipul și numărul interfeței de intrare/ieșire.
- *Next-hop* sau VCD. "*Next-hop*" specifică adresa IP a vecinului. "VCD" specifică numărul unui circuit virtual.
- *State/Mode*. "*State*": pachetele sau de tip prune sau null vor fi forwardate pe o interfață dacă sunt restricții printr-o listă de acces sau o limită TTL. "*Mode*" precizează dacă o interfață operează în modul *dense*, *sparse* sau *sparse-dense*.

Intrări în tabela multicast:

- (*,224.0.255.1), (198.92.37.100/32, 224.0.255.1): intrări in tabela de rutare multicast. Intrarea constă în adresa IP a ruterului sursă urmată de adresa IP a grupului multicast. Un asterisc (*) în locul ruterului sursă indică toate sursele. Intrările sunt de forma (S,G).
- *Uptime*: de cat timp (o:m:s) intrarea se află în tabela de rutare multicast.
- *Expires*: în cat timp (o:m:s) intrarea va fi eliminată din tabela de rutare multicast de pe interfața de ieșire.
- RP: adresa ruterului *rendevous point*. Pentru rutere care operează în *sparse mode*, această adresă va fi 0.0.0.0 .
- *Income interface*: interfața pe care se așteaptă un pachet de la sursă. Dacă pachetul este primit pe altă interfață, acesta este ignorat.
- RPF *neighbor*: adresa IP a ruterului către sursă. "*Tunneling*" indică faptul că acest ruter trimite date la RP încapsulate în pachete *Register*. Numărul hexa din paranteze indică cu ce RP se înregistrează.
- *Dvmrp* sau *Mroute*: indică dacă informația RPF a fost obținută de la tabela de rutare DVMRP sau prin rute configurate static – *mroute*.
- *Outgoing interface list*: Interfețele pe care vor fi trimise pachetele.
- *Next hop* sau VCD: specifică adresa IP a următorului ruter spre ascultători. VCD este numărul circuitului virtual.
- *Forward* / (*Dense* sau *Sparse*): indică faptul că pachetele vor fi trimise pe interfața de ieșire dacă nu sunt restricții datorate listelor de acces sau limitărilor TTL. După slash (/) se află modul în care interfața operează (sparse sau dense).

9.7.2 Interpretarea tabelei mroute

Intrarea care începe cu (*, 233.1.1.1) este o intrare din arborele multicast partajat. Mai este referită și ca (*,G). În cazul nostru, G este 233.1.1.1 – adresa grupului. PIM-DM nu folosește aceste intrări pentru a comuta pachete ci doar listează interfețele cu rol în multicast (primesc informații prin IGMP sau sunt vecini PIM) ca interfețe de ieșire.

Intrarea care începe cu (172.16.16.1, 233.1.1.1) este referită ca o intrare (S,G). S este sursa iar G este grupul multicast. Această intrare este un arbore multicast cu sursă specifică pentru un anumit grup multicast. Comanda va afișa câte o astfel de intrare pentru fiecare sursă și grup.

Se va observa faptul că următorul hop folosit în rutarea *unicast* apare ca vecinul RPF (*Reverse Path Forwarding*) și interfața de intrare apare ca interfață RPF. Această interfață (FastEthernet1) și următorul hop (10.20.30.1) sunt folosite de procesul de rutare pentru a trimite un pachet către sursă (172.16.16.1).

Interfețele de ieșire (în cazul nostru doar Ethernet0) indică faptul că orice pachet venit de la sursa 172.16.16.1, cu o adresă destinație multicast 233.1.1.1 și neapărat venite pe interfața FastEthernet1 a ruterului va fi trimis către Ethernet0 .

9.8 Studiu de caz

În studiul de caz ce urmează se va analiza operația realizată de un cercetător științific ce dorește să înregistreze statistici pe baza transferurilor multicast. Înainte de putea măsura însă performanțe și compara coeficienți de încărcare a rețelei, cercetătorul trebuie să configureze rețeaua pentru transmisii multicast. Administratorul de rețea al companiei în care se face acest

proiect i-a oferit cercetătorului o rețea IPv4 peste care rulează EIGRP. Urmează realizarea implementării multicast peste această rețea.

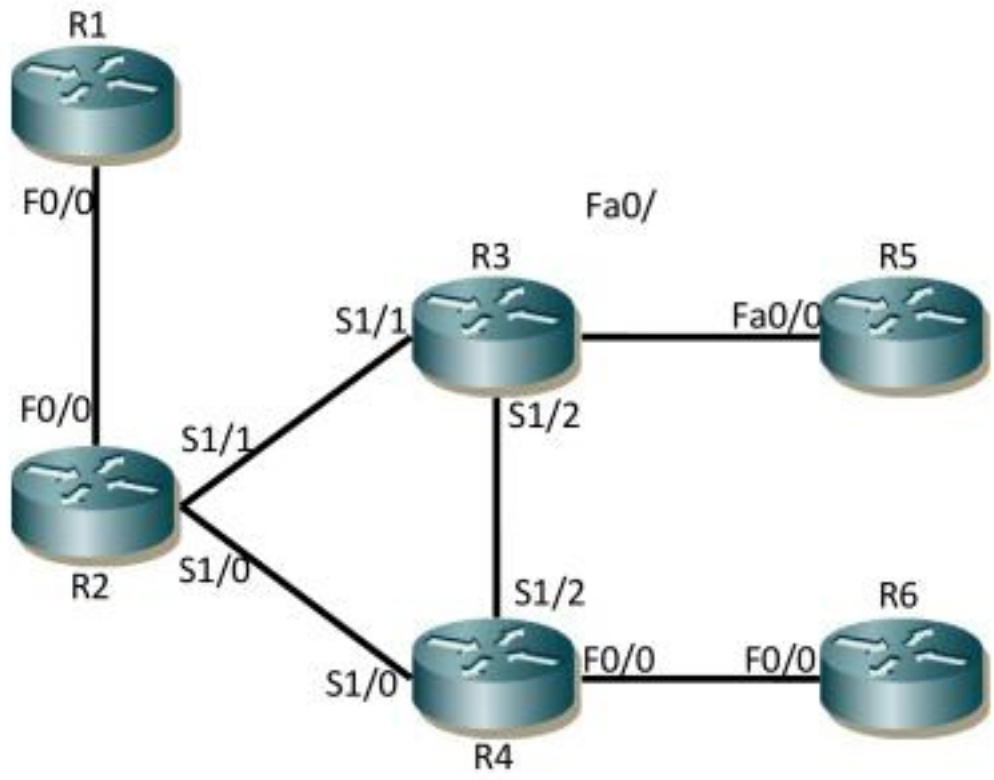

Figura 9-14: Studiu de caz

9.8.1 Configurarea PIM-DM

Mai întâi se dorește configurarea într -un protocol *dense mode*. Alegerea făcută de cercetător este protocolul PIM-DM. Folosind această opțiune el trebuie să trimită trafic multicast de la sursa R1 până la destinațiile R5 și R6. Aceste două rutere din urmă vor simula clienți de multicast pentru diferite grupuri folosind comanda **`ip igmp join-group`**.

Se va începe cu configurarea PIM-DM pe segmentele R2-R3, R2-R4, R3-R4 și pe interfețele Fa0/0 ale ruterelor R2, R3, R4. Interfețele F0/0 specificate trebuie activate în PIM pentru a putea realiza schimburi de mesaje cu clienții multicast R5 și R6. Deși este ușor neintuitivă activarea PIM pe o interfață pentru a putea schimba mesaje IGMP, acesta este modul în care IP Multicast a fost implementat în Cisco IOS.

```
R2(config)# ip multicast-routing
R2(config)# int Serial1/1
R2(config-if)# ip pim dense-mode
R2(config-if)# int Serial1/0
R2(config-if)# ip pim dense-mode
R2(config-if)# int Fa0/0
R2(config-if)# ip pim dense-mode

R3(config)# ip multicast-routing
R3(config)# int Serial1/1
R3(config-if)# ip pim dense-mode
R3(config-if)# int Serial1/2
R3(config-if)# ip pim dense-mode
R3(config-if)# int Fa0/0
R3(config-if)# ip pim dense-mode

R4(config)# ip multicast-routing
R4(config)# int Serial1/1
R4(config-if)# ip pim dense-mode
R4(config-if)# int Serial1/2
R4(config-if)# ip pim dense-mode
R4(config-if)# int Fa0/0
R4(config-if)# ip pim dense-mode
```

În continuare se va configura R6 pentru a primi trafic multicast pentru grupul 225.1.1.1.

```
R6(config)#int fa0/0
R6(config-if)#ip igmp join-group 225.1.1.1
```

Acum că infrastructura este pregătită pentru transmisia multicast în modul dense, cercetătorul va simula transferul de date pentru grupul 225.1.1.1 folosind un ping extins.

```
R1#ping
Protocol [ip]:
Target IP address: 225.1.1.1
Repeat count [1]: 5
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]:
Sweep range of sizes [n]:
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 225.1.1.1, timeout is 2 seconds:

Reply to request 0 from 10.1.46.6, 600 ms
Reply to request 1 from 10.1.46.6, 124 ms
Reply to request 2 from 10.1.46.6, 112 ms
Reply to request 3 from 10.1.46.6, 176 ms
Reply to request 4 from 10.1.46.6, 164 ms
```

Se poate observa faptul că *stream*-ul a fost transmis cu succes. Analizând *output*-ul comenzii **`show ip mroute`** pe ruterele din topologie, cercetătorul dorește să descopere drumul pe care pachetele de multicast l-au urmat în rețea.

Pe ruterul R2 se poate observa cum interfața serială 1/0 a fost pusă în modul *prune*. Motivul pentru acest rezultat este comportamentul implicit al protocolului PIM-DM de a trimite trafic pe toate interfețele. Când ruterul R3 a primit trafic multicast pentru grupul 225.1.1.1, pentru că nu există nicio stație direct conectată care să dorească traficul și niciun ruter *downstream* care să ceară trafic, acesta a trimis un mesaj un mesaj de *prune* vecinului său, R2.

```
R2#sh ip mroute
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C -
Connected,
       L - Local, P - Pruned, R - RP-bit set, F - Register flag,
       T - SPT-bit set, J - Join SPT, M - MSDP created entry,
       X - Proxy Join Timer Running, A - Candidate for MSDP
Advertisement,
       U - URD, I - Received Source Specific Host Report,
       Z - Multicast Tunnel, z - MDT-data group sender,
       Y - Joined MDT-data group, y - Sending to MDT-data group
Outgoing interface flags: H - Hardware switched, A - Assert winner
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 225.1.1.1), 00:06:06/stopped, RP 0.0.0.0, flags: D
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Serial1/0, Forward/Dense, 00:02:54/00:00:00
    Serial1/1, Forward/Dense, 00:03:07/00:00:00

(10.1.12.1, 225.1.1.1), 00:02:09/00:01:01, flags: T
  Incoming interface: FastEthernet0/0, RPF nbr 0.0.0.0
  Outgoing interface list:
    Serial1/1, Prune/Dense, 00:02:08/00:00:51
    Serial1/0, Forward/Dense, 00:02:09/00:00:00
```

Pe ruterul R3, interfața Serial1/2 a fost pusă în modul *pruned* însă dintr-un motiv complet diferit față de exemplul precedent. Când ruterul R3 a primit pentru prima oară trafic multicast de la

R2 a respectat comportamentul *dense* trimițând traficul pe toate interfețele sale. Astfel traficul multicast pentru 225.1.1.1 a ajuns de la R3 la R4. Când R4 a primit acest trafic, i-a aplicat testul RPF. Pentru că acest test a fost picat (calea cea mai scurtă de la R4 la sursă este prin R2, nu prin R3) ruterul R4 i-a trimis un mesaj de *prune* lui R3 specificându-i oprirea *stream*-ului. În acel moment, R3 a înțeles că nu există nimeni *downstream* care să dorească traficul trimis de el și a luat două acțiuni:

- A introdus interfața sa Serial1/2 în modul *pruned*;
- A trimis mesajul *prune* menționat în exemplul de mai sus către R2 spunându-i acestuia să nu mai trimită trafic.

```
R3#sh ip mroute
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C -
Connected,
       L - Local, P - Pruned, R - RP-bit set, F - Register flag,
       T - SPT-bit set, J - Join SPT, M - MSDP created entry,
       X - Proxy Join Timer Running, A - Candidate for MSDP
Advertisement,
       U - URD, I - Received Source Specific Host Report,
       Z - Multicast Tunnel, z - MDT-data group sender,
       Y - Joined MDT-data group, y - Sending to MDT-data group
Outgoing interface flags: H - Hardware switched, A - Assert winner
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 225.1.1.1), 00:01:10/stopped, RP 0.0.0.0, flags: D
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    Serial1/1, Forward/Dense, 00:01:10/00:00:00
    Serial1/2, Forward/Dense, 00:01:10/00:00:00

(10.1.12.1, 225.1.1.1), 00:01:10/00:02:02, flags: PT
  Incoming interface: Serial1/1, RPF nbr 10.1.23.2
  Outgoing interface list:
    Serial1/2, Prune/Dense, 00:01:10/00:01:53
```

Se va realiza acum atașarea ruterului R5 ca și destinatar pentru grupul 225.1.1.1:

```
R5(config)#int fa0/0
R5(config-if)#ip igmp join-group 225.1.1.1
```

Se pot observa schimbările din *output*-ul comenzilor **`sh ip mroute`**: ruterul R3 nu mai are interfața către R2 în modul *pruned* deoarece are un destinatar valid; cu toate acestea, interfața dinspre R4 este în continuare în modul *pruned* deoarece drumul ce se formează prin *link*-ul R3-R4 are un cost ineficient și pica testul RPF atât pe R3 cât și pe R4.

```
R3#sh ip mr
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C -
Connected,
       L - Local, P - Pruned, R - RP-bit set, F - Register flag,
       T - SPT-bit set, J - Join SPT, M - MSDP created entry,
       X - Proxy Join Timer Running, A - Candidate for MSDP
Advertisement,
       U - URD, I - Received Source Specific Host Report,
       Z - Multicast Tunnel, z - MDT-data group sender,
       Y - Joined MDT-data group, y - Sending to MDT-data group
Outgoing interface flags: H - Hardware switched, A - Assert winner
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 225.1.1.1), 00:03:59/stopped, RP 0.0.0.0, flags: DC
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    FastEthernet0/0, Forward/Dense, 00:00:35/00:00:00
```

```
     Serial1/1, Forward/Dense, 00:03:59/00:00:00
     Serial1/2, Forward/Dense, 00:03:59/00:00:00

(10.1.12.1, 225.1.1.1), 00:00:23/00:02:54, flags: T
  Incoming interface: Serial1/1, RPF nbr 10.1.23.2
  Outgoing interface list:
    Serial1/2, Prune/Dense, 00:00:23/00:02:39, A
    FastEthernet0/0, Forward/Dense, 00:00:23/00:00:00
```

9.8.2 Configurarea PIM-SM

După realizarea măsurătorilor pentru PIM-DM, cercetătorul în cauză a trecut la analiza unui protocol *sparse*. Deoarece comportamentul de realizare a arborilor de acoperire multicast este diferit în cele două cazuri era clară nevoia configurării unei infrastructuri PIM-SM pentru ca testele de performanță să fie relevante.

Pentru început, toate interfețele sunt trecute din PIM *dense* în PIM *sparse* folosind comanda **`ip pim sparse-mode`**. Pentru a crea o topologie pe care comportamentul *SPT Switchover* să poată fi ușor observabil, cercetătorul a închis temporar interfața dintre R2 și R4. De asemenea R3 a fost ales pentru rolul de RP din rețea. Configurarea acestuia va fi făcută static pe toate dispozitivele din rețea folosind ca și adresă de RP, o adresă a unui *loopback* a lui R3. Se va folosi adresa de *loopback* ca RP deoarece aceasta este o interfață virtuală permanent activă. Prima regulă a transmiterii de multicast este însă conectivitatea *end-to-end* IPv4, astfel că cercetătorul este obligat să introducă rețeaua de *loopback* în protocolul EIGRP.

```
! configurarea loopback-ului
R3#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
R3(config)#int lo0
R3(config-if)#ip add
*Mar  1 00:45:07.395: %LINEPROTO-5-UPDOWN: Line protocol on Interface
Loopback0, changed state to up
R3(config-if)#ip address 150.1.1.1 255.255.255.0
R3(config-if)#exit
R3(config)#router eigrp 100
R3(config-router)#net
R3(config-router)#network 150.1.1.0 0.0.0.255
R3(config-router)#exit
```

```
! schimbarea din dense-mode în sparse mode; această schimbare trebuie
! făcută inclusiv pe R2  i R3 chiar dacă acest lucru este omis din
! exemplu
R4(config)#int serial 1/0
R4(config-if)#ip pim sparse-mode
R4(config-if)#int ser1/2
R4(config-if)#ip pim sparse- mode
R4(config-if)#int fa0/0
R4(config-if)#ip pim sparse- mode
```

```
! configurarea statică a adresei RP-ului
R2(config)#ip pim rp-address 150.1.1.1
R4(config)#ip pim rp-address 150.1.1.1
R3(config)#ip pim rp-address 150.1.1.1
```

După cum se observă mai sus, configurarea RP-ului trebuie făcută inclusiv pe RP pentru ca *sparse-mode* să funcționeze. Deoarece se folosește PIM-SM, o interfață nu va mai apărea ca fiind *pruned* în tabela de rutare multicast, ci pur și simplu nu va mai apărea dacă nu se primește nici un mesaj de *join-tree*.

```
R3#sh ip mroute
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C -
Connected,
       L - Local, P - Pruned, R - RP-bit set, F - Register flag,
       T - SPT-bit set, J - Join SPT, M - MSDP created entry,
       X - Proxy Join Timer Running, A - Candidate for MSDP
Advertisement,
       U - URD, I - Received Source Specific Host Report,
       Z - Multicast Tunnel, z - MDT-data group sender,
       Y - Joined MDT-data group, y - Sending to MDT-data group
Outgoing interface flags: H - Hardware switched, A - Assert winner
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 225.1.1.1), 00:21:14/00:03:18, RP 150.1.1.1, flags: SJC
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    FastEthernet0/0, Forward/Sparse, 00:17:50/00:02:49
    Serial1/1, Forward/Sparse, 00:21:14/00:00:00
    Serial1/2, Forward/Sparse, 00:21:14/00:03:18

(10.1.12.1, 225.1.1.1), 00:00:36/00:02:37, flags: T
  Incoming interface: Serial1/1, RPF nbr 10.1.23.2
  Outgoing interface list:
    FastEthernet0/0, Forward/Sparse, 00:00:36/00:02:48
```

Pentru a analiza operația de *SPT Switchover* care este ultimul concept pe care cercetătorul dorește să îl testeze, se va analiza mai întâi comanda **`sh ip mroute`** pe ruterul R4.

```
R4#sh ip mroute
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C -
Connected,
       L - Local, P - Pruned, R - RP-bit set, F - Register flag,
       T - SPT-bit set, J - Join SPT, M - MSDP created entry,
       X - Proxy Join Timer Running, A - Candidate for MSDP
Advertisement,
       U - URD, I - Received Source Specific Host Report,
       Z - Multicast Tunnel, z - MDT-data group sender,
       Y - Joined MDT-data group, y - Sending to MDT-data group
Outgoing interface flags: H - Hardware switched, A - Assert winner
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 225.1.1.1), 00:51:23/stopped, RP 150.1.1.1, flags: SJC
  Incoming interface: Serial1/2, RPF nbr 10.1.34.3
  Outgoing interface list:
    FastEthernet0/0, Forward/Sparse, 00:51:23/00:02:51

(10.1.12.1, 225.1.1.1), 00:00:06/00:02:55, flags: JT
  Incoming interface: Serial1/2, RPF nbr 10.1.34.3
  Outgoing interface list:
    FastEthernet0/0, Forward/Sparse, 00:00:06/00:02:53
```

Se poate observa din adresa de RPF a vecinului (RPF nbr 10.1.34.3) faptul că tot traficul multicast trece de la R4 la R3și invers. În acest moment se va da **`no shutdown`** pe legătura dintre R2 și R4 unde EIGRP este deja pornit. PIM-SM va folosi metricile EIGRP pentru a-și da seama că drumul prin R2 este mai bun acum decât drumul prin R3. Astfel va trimite *prune* către arborele *shared* și se va atașa unui arbore SPT cu sursa de multicast ca rădăcină.

```
R4#sh ip mroute
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group, C -
Connected,
       L - Local, P - Pruned, R - RP-bit set, F - Register flag,
       T - SPT-bit set, J - Join SPT, M - MSDP created entry,
       X - Proxy Join Timer Running, A - Candidate for MSDP
Advertisement,
       U - URD, I - Received Source Specific Host Report,
       Z - Multicast Tunnel, z - MDT-data group sender,
       Y - Joined MDT-data group, y - Sending to MDT-data group
Outgoing interface flags: H - Hardware switched, A - Assert winner
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 225.1.1.1), 00:46:10/stopped, RP 150.1.1.1, flags: SJC
  Incoming interface: Serial1/2, RPF nbr 10.1.34.3
  Outgoing interface list:
    Serial1/0, Forward/Sparse, 00:32:15/00:00:00
    FastEthernet0/0, Forward/Sparse, 00:46:10/00:01:58

(10.1.12.1, 225.1.1.1), 00:00:13/00:02:57, flags: JT
  Incoming interface: Serial1/0, RPF nbr 10.1.24.2
  Outgoing interface list:
    FastEthernet0/0, Forward/Sparse, 00:00:13/00:02:46
```

Cuprins

1 Arhitectura unui ruter 1

1.1 Componentele unui ruter 1
1.1.1 Memoriile unui ruter 2
1.1.2 Interfețele unui ruter 3
1.1.3 Procesul de inițializare al unui ruter 4
1.2 Sistemul de operare 5
1.2.1 Moduri de configurare 5
1.2.2 Sistemul de fişiere 6
1.3 Configurații de bază 6
1.3.1 Protejarea accesului la ruter 8
1.3.2 Configurarea unei interfețe 11
1.3.3 Verificarea configurațiilor de bază 14
1.3.4 Facilități ale liniei de comandă 17

2 Funcțiile unui ruter 21

2.1 Rutarea 21
2.1.1 Tabela de rutare 22
2.1.2 Rutarea statică 23
2.1.3 Protocoale de rutare 24
2.2 RIP 27
2.2.1 Descrierea protocolului 27
2.2.2 Configurarea protocolului 28
2.2.3 Caracteristici ale protocoalelor distance-vector 30
2.2.4 RIPv2 31
2.3 Liste de acces (ACLs) 32
2.3.1 Modul de funcționare a listelor de acces 33
2.3.2 Tipuri de liste de acces 34
2.3.3 Configurarea listelor de acces standard 34
2.3.4 Configurarea listelor de acces extinse 35
2.3.5 Configurarea listelor de acces cu nume 36
2.3.6 Ordonarea şi documentarea listelor de acces 37
2.4 Translatarea de adrese 38
2.4.1 Adrese publice şi private 38
2.4.2 Prezentarea conceptelor 38
2.4.3 Configurarea NAT 40
2.4.4 Configurare PAT 41
2.4.5 Port forwarding 42

3 IPv6 43

3.1 Protocolul IPv6 44
3.1.1 Antetul IPv6 44
3.1.2 Reprezentarea adreselor IPv6 45
3.1.3 Structura adreselor IPv6 46
3.1.4 Network Discovery Protocol 48

3.2 Implementarea IPv6 50
3.2.1 Rutarea IPv6 50
3.2.2 Interconectarea rețelelor IPv4 cu rețele IPv6 50
3.3 Configurarea IPv6 53
3.3.1 Configurarea adresării 53
3.3.2 Configurări de rutare 54
3.3.3 Configurări de interconectare IPv4-IPv6 55
3.4 Studiu de caz 56
4 EIGRP 59
4.1 Prezentarea protocolului 59
4.1.1 Caracteristici ale EIGRP 59
4.1.2 Descoperirea vecinilor EIGRP 60
4.1.3 Calculul metricii EIGRP 60
4.1.4 Mașina de stări finite DUAL 61
4.1.5 Tabelele EIGRP 62
4.1.6 Formatul pachetelor EIGRP 63
4.2 Concepte avansate 64
4.2.1 Sumarizarea rutelor 64
4.2.2 Echilibrarea traficului 64
4.2.3 Scalabilitatea EIGRP în rețele mari 64
4.2.4 EIGRP Queries 65
4.2.5 EIGRP Stub 65
4.2.6 Autentificarea EIGRP 65
4.3 Configurarea protocolului 66
4.3.1 Configurări de bază 66
4.3.2 Propagarea rutei implicite 68
4.3.3 Verificarea funcționării EIGRP 68
4.3.4 Configurarea sumarizării 72
4.3.5 Configurarea balansării traficului 73
4.3.6 Configurarea ruterului stub 74
4.3.7 Configurarea autentificării 75
4.4 Studiu de caz 78
4.4.1 Configurări de bază 79
4.4.2 Optimizarea timpilor de convergență folosind DUAL 80
4.4.3 Optimizarea timpilor de convergență folosind rețele Stub 82
4.4.4 Minimizarea dimensiunii tabelei de rutare 82
5 OSPF 85
5.1 Prezentarea protocolului 85
5.1.1 Link state vs. distance vector 85
5.1.2 Funcționarea protocolului 87
5.1.3 Tabelele OSPF 90
5.1.4 Formatul pachetelor OSPF 91
5.1.5 OSPF Multi-area 92
5.1.6 Tipuri de LSA 94
5.2 Concepte avansate 96

5.2.1 Tipuri de zone 96
5.2.2 Sumarizarea rutelor 99
5.2.3 Autentificarea în OSPF 99
5.2.4 Legături virtuale 99
5.3 Configurarea protocolului 100
5.3.1 Configurări de bază 100
5.3.2 Verificarea funcționării OSPF 102
5.3.3 Configurarea unei zone Stub 104
5.3.4 Configurare zonă Totally Stubby 105
5.3.5 Configurare zonă Not So Stubby 106
5.3.6 Configurarea sumarizării 107
5.3.7 Configurarea autentificării 109
5.3.8 Configurarea legăturilor virtuale 112
5.4 Studiu de caz 114
5.4.1 Topologie 114
5.4.2 Configurări OSPF Single-area 115
5.4.3 Configurare OSPF Multi-area 121
6 IS-IS 131
6.1 Componentele protocolului 131
6.1.1 ISO vs. IETF 131
6.1.2 Rutarea cu IS-IS 131
6.1.3 IS-IS vs. OSPF 132
6.1.4 Adresarea ISO 134
6.2 Funcționarea IS-IS 135
6.2.1 Mesaje 135
6.2.2 Adiacențe 136
6.2.3 Backbone L2 137
6.2.4 Funcționare L1 137
6.2.5 Funcționarea L12 138
6.3 Configurarea protocolului 138
6.3.1 Comenzi de verificare 140
6.4 Studiu de caz 142
6.4.1 Configurări de bază 143
6.4.2 Controlul procesului de alegere DIS 145
6.4.3 Alterarea metricilor 147
6.4.4 Autentificarea în IS-IS 147
7 Optimizarea rutării 149
7.1 Redistribuția între multiple protocoale de rutare 149
7.1.1 Conceptul de redistribuție 150
7.1.2 Problema rutării suboptimale 151
7.1.3 Bucle de rutare 154
7.2 Manipularea rutării 154
7.2.1 Manipularea *update*-urilor de rutare 154
7.2.2 Policy-based routing 157
7.3 Configurări în optimizarea rutării 159

7.3.1 Configurarea redistribuirii între protocoale de rutare 159
7.3.2 Configurarea filtrării rutelor 161
7.3.3 Configurarea PBR 161
7.4 Studiu de caz 162
7.4.1 Configurări inițiale 163
7.4.2 Redistribuirea protocoalelor de rutare 165
7.4.3 Probleme la redistribuție 166
7.4.4 Policy Based Routing 167
8 BGP 169
8.1 Protocolul BGP 169
8.1.1 Sisteme autonome 169
8.1.2 Când este necesar BGP? 170
8.1.3 BGP - noțiuni de bază 170
8.1.4 iBGP vs eBGP 171
8.1.5 Next-hop 172
8.1.6 Tabele BGP 173
8.2 Implementarea politicilor BGP 173
8.2.1 Atribute BGP 173
8.2.2 Procesul de decizie BGP 176
8.2.3 Scalarea BGP 177
8.3 Configurarea protocolului 178
8.3.1 Configurarea adiacențelor BGP 178
8.3.2 Configurarea adiacențelor eBGP 180
8.3.3 Configurarea politicilor BGP 182
8.3.4 Interpretarea tabelelor de topologie BGP 183
8.4 Studiu de caz 183
8.4.1 Configurații de bază 184
8.4.2 Manipularea atributelor BGP 189
9 IP Multicast 193
9.1 De ce avem nevoie de multicast ? 193
9.2 Funcționarea multicast 194
9.3 Adresarea multicast 196
9.3.1 Structura adreselor multicast 196
9.3.2 Maparea adreselor multicast IP la adrese MAC 197
9.4 Gestiunea grupurilor multicast în LAN 198
9.4.1 IGMP v2 198
9.4.2 Antetul IGMPv2 198
9.4.3 Înregistrarea la un grup de multicast 199
9.4.4 Părăsirea unui grup de multicast 202
9.5 Optimizări ale transmiterii multicast în LAN 203
9.5.1 Cisco Group Management Protocol 204
9.5.2 IGMP Snooping 204
9.6 Rutarea multicast 205
9.6.1 PIM-DM 205
9.6.2 Reverse Path Forwarding Check 206
9.6.3 Source-Based Trees 207

9.6.4 Mesajul *graft* 208
9.6.5 PIM-SM 208
9.6.6 Funcționarea PIM-SM: SPT și RPT 209
9.6.7 Optimizarea PIM-SM: SPT switchover 210
9.7 Configurări de bază 211
9.7.1 Parametrii 212
9.7.2 Interpretarea tabelei mroute 213
9.8 Studiu de caz 213
9.8.1 Configurarea PIM-DM 214
9.8.2 Configurarea PIM-SM 217

ISBN 978-606-521-092-9

Material cu drepturile autor